

Plant Identification Using Deep Convolutional Networks
Based on Principal Component Analysis

by
Mostafa Mehdipour Ghazi

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Master of Science

SABANCI UNIVERSITY

August 2015

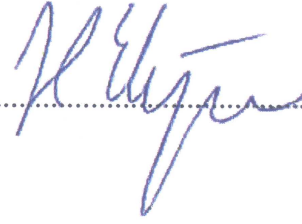
Plant Identification Using Deep Convolutional Networks Based on Principal Component Analysis

APPROVED BY

Assoc. Prof. Dr. Berrin Yanıkoğlu
(Thesis Supervisor)



Assoc. Prof. Dr. Hakan Erdoğan



Assoc. Prof. Dr. Erchan Aptoula



DATE OF APPROVAL: 2015-08-05

© Mostafa Mehdipour Ghazi 2015
All Rights Reserved

To my family

Acknowledgements

My sincere gratitude goes to my supervisor Prof. Berrin Yanıkoğlu for providing the opportunity to work alongside her on this exciting and extremely challenging project. She trusted and supported me during my endeavor for exploring the vast field of machine learning, and provided motivation, guidance, patience, and immense knowledge throughout this journey. I should also seize the opportunity to thank her for teaching the highly interesting and beneficial course of Deep Learning together with Prof. Hakan Erdoğan.

Furthermore, I would like to extend my gratitude to Prof. Hakan Erdoğan for being a jury member of my thesis and sharing his valuable suggestions for improving the quality of this work. Prof. Erdoğan also patiently supervised me during my first year at Sabancı University and has been a great source of friendship, support, and encouragement throughout my studies. I am also grateful for his teaching of the Random Process course and his supervision during my independent work on image noise level estimation.

I also thank to Prof. Erchan Aptoula for being on my thesis defense committee and for his invaluable collaboration and suggestions during the LifeCLEF plant identification competition.

Thanks to Prof. Aytul Erçil for offering extremely useful courses of Computer Vision and Pattern Recognition which came of great use during this work. She has always been patient and kind to me and supportive of my personal growth.

I would like to offer my heart-felt gratitude to my family for their constant and unconditional love and encouragement always and especially during my stay in Istanbul. I would also like to thank all VPALAB members especially Fahad Sohrab, Ismail Yılmaz, Amir Abbas Davari, and Rahim Dehkharghani for their friendship and sharing nice memories. Last but not least, my especial thanks go to Mastaneh Torkamani for her truly valued help and support for this work.

This work has been generously supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under the grant number 113E499.

Abstract

Plant Identification Using Deep Convolutional Networks Based on Principal Component Analysis

Mostafa Mehdipour Ghazi

EE, M.Sc. Thesis, August, 2015

Thesis Supervisor: Prof. Berrin Yanikoglu

Keywords: object recognition, plant identification, principal component analysis, deep convolutional networks, spatial pyramid pooling.

Plants have substantial effects in human vitality through their different uses in agriculture, food industry, pharmacology, and climate control. The large number of herbs and plant species and shortage of skilled botanists have increased the need for automated plant identification systems in recent years. As one of the challenging problems in object recognition, automatic plant identification aims to assign the plant in an image to a known taxon or species using machine learning and computer vision algorithms. However, this problem is challenging due to the inter-class similarities within a plant family and large intra-class variations in background, occlusion, pose, color, and illumination.

In this thesis, we propose an automatic plant identification system based on deep convolutional networks. This system uses a simple baseline and applies principal component analysis (PCA) to patches of images to learn the network weights in an unsupervised learning approach. After multi-stage PCA filter banks are learned, a simple binary hashing is applied to output maps and the obtained maps are subsampled through max-pooling. Finally, the spatial pyramid pooling is applied to the downsampled data to extract features from block histograms. A multi-class linear support vector machine is then trained to classify the different species.

The system performance is evaluated on the plant identification datasets of LifeCLEF 2014 in terms of classification accuracy, inverse rank score, and robustness against pose (translation, scaling, and rotation) and illumination variations. A comparison of our results with those of the top systems submitted to LifeCLEF 2014 campaign reveals that our proposed system would have achieved the second place in the categories of Entire, Branch, Fruit, Leaf, Scanned Leaf, and Stem, and the third place in the Flower category while having a simpler architecture and lower computational complexity than the winner system(s). We achieved the best accuracy in scanned leaves where we obtained an inverse rank score of 0.6157 and a classification accuracy of 68.25%.

Özet

Ana Bileşen Analizine Dayalı Derin Konvolüsyonel Ağ Kullanımıyla Bitki Tanımlama

Mostafa Mehdipour Ghazi

Elektronik Mühendislik, Yüksek Lisans Tezi, Ağustos, 2015

Tez Danışmanı: Prof. Berrin Yanıkoğlu

Anahtar Kelimeler: nesne tanıma, bitki tanımlama, uzamsal piramit birleştirmesi, ana bileşen analizi, derin konvolüsyonel ağ.

Gıda endüstrisi, tarım, farmakoloji ve iklim kontrolü gibi çeşitli alanlardaki kullanımıyla, bitkiler insan yaşamı bakımından çok önemlidir. Ot ve bitki türlerinde muazzam bir çeşitlilik görülmesi, üstelik yeterli niteliklere sahip botanistlerin sayıca bir hayli az olması nedeniyle son yıllarda otomatik bitki tanımlama sistemlerine duyulan ihtiyaç artmıştır. Nesne tanıma teknolojisindeki en zor sorunlardan birine çözüm getirmeyi amaçlayan otomatik bitki tanımlama, otomatik öğrenme ve bilgisayarla görme algoritmalarını kullanarak bir görselde yer alan bitkiyi bilinen text veya türe atamayı hedefler. Ancak tanıma işlemi bitki ailelerindeki sınıflararası benzerlikler ve arka plan, örtme, poz, renk ve aydınlatmadaki sınıf içi varyasyonlar nedeniyle zorlaşır.

Bu tezde, derin konvolüsyonel ağ bazlı otomatik bitki tanımlama sistemi çözümü önerilmektedir. Gözetimsiz öğrenim yaklaşımına dayanan sistem, basit bir temel kullanarak görsel parçalarına Ana Bileşenl Analizi (ABA) uygulayıp ağ ağırlıklarını öğrenir. Çok aşamalı ABA filtre öbekleri öğrenildikten sonra, çıkış haritalarında basit ikili kıyım gerçekleştirilir. Ardından haritalarda maksimum havuzlama ile altörneklem elde edilir. Son olarak altörneklem ile elde edilen verilere uzamsal piramit birleştirmesi uygulanarak blok histogramdan özellik detayları çıkarılır. Bunun ardından, çok sınıflı lineer destek

vektör makinesi farklı türleri sınıflandırmak üzere eğitilir.

Sistem performansı, LifeCLEF 2014 bitki tanımlama veri kümeleri üzerinde sınıflandırma doğruluğu ve ters sıralama puanına ek olarak, poz (translasyon, ölçeklendirme, ve döndürme) ve aydınlatma varyasyonlarına karşı dayanıklılık bakımından değerlendirilmiştir. Elde ettiğimiz sonuçlar, LifeCLEF 2014 kampanyasına gönderilen en iyi sistemlerde elde edilen sonuçlar ile karşılaştırıldığında; Genel, Dal, Meyve, Yaprak, Taranmış Yaprak ve Kök kategorilerinde ikinci, Çiçek kategorisinde ise üçüncü sırayı denk gelmektedir; üstelik birinci sırayı alan sistem(ler)e kıyasla daha basit bir mimari kullandığımız ve hesaplama karmaşıklığının da daha düşük olduğu görülmektedir. En yüksek doğruluk oranını ise 0,6157 ters sıralama puanı ve 68,25% sınıflandırma doğruluğu elde ettiğimiz taranmış yaprak kategorisinde yakaladığımız anlaşılmıştır.

Contents

Acknowledgement	iv
Abstract	v
Özet	vii
List of Figures	xii
List of Tables	xiv
1 Introduction	1
1.1 Objectives	3
1.2 Limitations	4
1.3 Thesis Structure	5
2 The Plant Identification Problem	7
2.1 Object Recognition	7
2.2 Plant Identification	8
2.2.1 Image Acquisition and Preprocessing	10
2.2.2 Common Methods for Leaf Analysis	11
2.2.2.1 Shape Analysis	12
2.2.2.2 Texture Analysis	15
2.2.2.3 Venation Analysis	15
2.2.2.4 Segmentation	15
2.2.3 Common Methods for Flower Analysis	16
2.2.4 Highlights of Plant Identification Systems in CLEF Campaigns	17
3 Convolutional Neural Networks	20
3.1 Deep Learning	20
3.2 Artificial Neural Networks	22

3.2.1	Historical Background	22
3.2.2	Model of Biological Neuron	23
3.2.3	Perceptron	24
3.2.4	Activation Function	25
3.2.5	Neural Networks Architecture	26
3.2.5.1	Feedforward Networks	27
3.2.5.2	Feedback Networks	27
3.2.6	Learning Process	28
3.2.6.1	Supervised Learning	29
3.2.6.2	Unsupervised Learning	30
3.2.6.3	Backpropagation Algorithm	30
3.3	Convolutional Neural Networks	32
3.3.1	CNN Architecture	32
3.3.1.1	Convolutional Layer	33
3.3.1.2	Pooling Layer	34
3.3.1.3	Normalization Layer	35
3.3.1.4	Output Layer	35
4	Object Recognition Using Deep Convolutional Networks Based on PCA	38
4.1	Background	38
4.2	Motivations	39
4.3	Contributions	40
4.4	Proposed Deep PCA Network	40
4.4.1	Spatial Pyramid Pooling	43
4.4.2	Classification by Linear SVM	44
5	Experiments and Results	47
5.1	Dataset Description	47
5.2	Preprocessing	48
5.3	Evaluation Metrics	49
5.4	Experimental Methods	50
5.4.1	Effects of Parameter Adjustment	50
5.4.1.1	Number of Learning Stages	51

5.4.1.2	Number of Filters	51
5.4.1.3	Filtering Patch Size	52
5.4.1.4	Spatial Pyramid Levels	52
5.4.1.5	Image Size Normalization	53
5.4.2	Classification Results	53
5.4.3	Robustness of the Proposed System	54
5.4.3.1	Scale Invariability	55
5.4.3.2	Translation Invariability	55
5.4.3.3	Rotation Invariability	55
5.4.3.4	Illumination Invariability	56
5.5	Time Complexity	56
5.6	Learned Filter Banks	56
6	Summary and Conclusion	61
A	HSY Color Space	63
B	Acronyms	65
	Bibliography	68

List of Figures

2.1	The main features and botanical terms of a typical leaf	11
2.2	Shape features for a typical leaf ROI	13
2.3	An example of elliptic Fourier analysis	14
3.1	The structure of a biological neuron	24
3.2	Model of a perceptron	25
3.3	A fully-connected three-layer neural network	26
3.4	A two-layer fully-connected feedforward neural network	27
3.5	An example of feedback (recurrent) neural network	28
4.1	Block diagram of a two-stage PCA network	46
5.1	Samples of LifeCLEF 2014 plant dataset	48
5.2	Effects of preprocessing on scanned leaf images	49
5.3	Learned weights from Branch in the first stage	57
5.4	Learned weights from Branch in the second stage	57
5.5	Learned weights from Entire in the first stage	57
5.6	Learned weights from Entire in the second stage	57
5.7	Learned weights from Flower in the first stage	58
5.8	Learned weights from Flower in the second stage	58
5.9	Learned weights from Fruit in the first stage	58
5.10	Learned weights from Fruit in the second stage	58
5.11	Learned weights from Leaf in the first stage	59
5.12	Learned weights from Leaf in the second stage	59
5.13	Learned weights from LeafScan in the first stage	59
5.14	Learned weights from LeafScan in the second stage	59
5.15	Learned weights from preprocessed LeafScan in the first stage	60

5.16	Learned weights from preprocessed LeafScan in the second stage	60
5.17	Learned weights from Stem in the first stage	60
5.18	Learned weights from Stem in the second stage	60

List of Tables

5.1	Details of plant identification datasets within the LifeCLEF 2014	47
5.2	Classification results for different number of learning stages	51
5.3	Classification results for different number of filters in the first stage	51
5.4	Classification results for different number of filters in the second stage	52
5.5	Classification results for different filtering patch sizes	52
5.6	Classification results for different levels of spatial pyramid	53
5.7	Classification results for different image sizes	53
5.8	Classification results of the proposed method in LifeCLEF 2014	54
5.9	Inverse rank scores of different systems submitted to the LifeCLEF 2014	54
5.10	Classification results for different scales of test images	55
5.11	Classification results for different translation sizes of test images	55
5.12	Classification results for different rotation angles of test images	55
5.13	Classification results for different intensities of test images	56

Chapter 1

Introduction

Visual recognition, the process of recognizing shapes and their properties through visual observation, is a complex yet well-developed ability of the human brain. Humans can detect and distinguish among over 30,000 visual categories in various situations arising from different viewpoints, illuminations, or occlusions [1]. Indeed, human brain uses a high-level, perceptual organization for object recognition; i.e. it realizes that 3D objects look different from various viewpoints and considers the invariance of features such as connectivity, texture, and symmetries as a result of projection [2]. Due to its complexity and computationally demanding nature, object recognition remains an open problem for neuroscientists [3].

Besides being a topic of interest for cognitive neuroscience context, object recognition through vision is a heavily investigated problem in the field of computer vision as well. It is generally defined as the detection and identification of objects within sequences of still or moving images, and is further divided into two tasks of identification and categorization. In order to develop robust, efficient and fast automatic object recognition systems, attempts have been made to utilize the color information [4], reflectance properties [5], and model information [6] from objects. However, these methods are able to yield high accuracy in only single-object identification including large inter-class dissimilarity and low intra-class variability [7]. That is to say, computer-based vision techniques are better in identification of objects than their categorization as the latter requires having access to a large database of attributes as well as their hierarchical and interleaved relations [8].

One of the challenging tasks of object recognition that has attracted increasingly more interest in the field of computer vision is plant identification. Identification and later

classification of plants is of course important in the fields of botany, agriculture, plant taxonomy, and pharmacology. Nevertheless, the large number of herbs and plant species and shortage of skilled botanists have increased the need for developing automated plant identification systems for computers and mobile devices to identify various organs of earth flora [9].

In recent years, research in the area of automatic plant identification from photographs has concentrated around annual plant identification competitions that are organized within campaigns of the Conference and Labs of the Evaluation Forum (CLEF) including ImageCLEF [10–12] and LifeCLEF [13, 14]. CLEF is devoted to promoting and evaluating multilingual and multimodal information retrieval systems and the main goal of these competitions is to benchmark the challenging task of content-based identification and retrieval of plant species from structured databases of their parts including leaves, branches, stems, flowers, and fruits.

Content-based image retrieval (CBIR) is a key idea challenged through the image-based and observation-based tasks of CLEF campaigns which investigate image queries in a large database by analyzing image contents such as colors, shapes, textures, or any other information that can be derived from the image [15]. Specifically, CBIR systems developed for plant identification applications have focused on exploiting shape, texture, and contour information as discriminant features. Color information, on the other hand, has been shown to be less efficient especially for leaf identification since most plant species have green shades and their color may change throughout the year [16]. However, the greatest challenge in plant identification has been the large variations in background, occlusion, illumination, pose (translation, orientation, and scaling) which causes plant identification problems suffer from intra-class variations and inter-class similarities more than other object recognition tasks [7]. Therefore, extracting simple and low-level features from domains such as shape, texture, and color fail to provide robust identification results. In this respect, deep learning approaches are new and offer a suitable solution for such complex problems.

Contrary to traditional machine learning methods where the features are chosen manually and extracted through instructed algorithms, deep learning methods such as convolutional neural networks (CNNs/ConvNets) and deep belief networks (DBNs) feed raw data into the system in multiple levels and allow it to automatically discover low-level and

high-level features or representations that can be used for detecting, distinguishing, and classifying patterns [17]. Still, these systems suffer from high computational complexity due to using optimization techniques to learn multi-stage weights.

The first mathematically-driven deep network architecture using prefixed weights were scattering networks (ScatNets) [18, 19]. They were applied for texture discrimination and showed superior performance over CNNs; but due to their prefixed nature, ScatNets could not be generalized well to problems with large intra-class variance such as face recognition or plant identification. To tackle this issue, PCA network (PCANet) [20], a hybrid of principal component analysis and deep convolutional networks (DCNs) was proposed and tested successfully in such problems. This system learns weights in an unsupervised learning manner similar to a DBN with no feedback and applies learned filter banks in a CNN-like way. It offers noise and dimensionality reduction and confronts with overfitting issues.

The system we propose in this study manages to combine the best of both worlds while providing more pose invariance, reduction in overfitting, and utilization of color information in images. This system is tested over LifeCLEF 2014 plant identification datasets and compared with the participants in the same campaign. Ten participating teams submitted 27 runs or systems in total to the organizers of LifeCLEF 2014 and, as the results presented in Section 5.4.2 show, our proposed system would have achieved almost the second place among the top six teams of this competition. It is noteworthy to mention that our learned model has the advantage of being architecturally and computationally simpler compared to the deep convolutional neural network system proposed by the winner of this competition.

1.1 Objectives

Several applications of plant identification in botany, pharmacology, agriculture, ecological preservation programs, and the like have been the driving forces behind the demand for developing automated plant identification systems. These applications require systematic activities for acquiring images, creating informative databases, performing preprocessing, extracting low-level and high-level information, and classifying using computer vision and machine learning techniques.

Regardless of applying either manual or automatic feature extraction techniques, feature extraction techniques, plant identification faces problems within real databases including large intra-class variations. For instance, photographing different plant organs including leaves, flowers, fruits, stems, and branches in natural environments introduces issues such as partial occlusions of organs of interest by other plants or objects, various poses of organs, color fading due to seasonal changes, and illumination variations due to daylight, shadow, etc. These problems make samples of the same species or category look different for a computer vision and machine learning system.

The first objective of this study was to design a simple and robust object recognition system that is able to tackle issues related to high intra-class variabilities, especially applicable to plant identification problems that have been systematically organized within the CLEF campaigns and competitions. Image datasets provided for these competitions are collected by different photographers and users in natural settings with various illuminations and pose conditions to provide a large degree of intra-class variance. They currently include hundreds of thousands of images from around 1,000 species of trees and herbaceous plants [14].

Since unsupervised-learning-based deep convolutional networks have already obtained superior results in such challenging identification tasks, our key objective is to design a system in this field that provides reduced architecture and computational complexity—two major issues that arise when dealing with huge datasets.

1.2 Limitations

Besides facing common problems of plant identification including inter-class similarities and intra-class variabilities, CLEF datasets contain a large dataset including hundreds of thousands of images [13, 14]. Considering the competitive nature of this task, designing and implementing deep convolutional networks that have high classification accuracy require high computational loads for determining weights and other model parameters. These issues impose architectural complexity in the number of layers as well as the learning time issue.

Another limitation we faced was related to image preprocessing before applying machine learning schemes. For instance, we preprocessed scanned leaves by segmentation,

background removal, and size and orientation normalization. As we will see in Section 5.4.2, preprocessing dramatically increases the system performance. This suggests that preprocessing using computer vision and image processing algorithms should be a prerequisite for the proposed deep learning system. However, we could not perform any preprocessing on images from other categories due to time limitations. We could have evaluated the system performance by finding the region of interest, performing size and orientation normalization and background removal, and omitting unnecessary elements such as the petiole had the time allowed.

1.3 Thesis Structure

The rest of this thesis is organized as follows. Chapter 2 provides an introduction to generic (category-level) object recognition as well as motivations for performing plant identification. It contains a review of shape, texture, and other key feature analysis algorithms for identifications of plants in general and leaves in particular. The chapter ends with an overview of approaches from top-ranking participants of plant identification tasks in ImageCLEF 2012 and LifeCLEF 2014.

Chapter 3 includes an overview of deep learning specifically based on convolutional neural networks. It describes the core concepts of deep neural networks (DNNs), common architectures and properties of general artificial neural networks (ANNs) as well as concepts and motivations behind CNN structures. The final section of this chapter offers a thorough overview and visualization of layers and building blocks of CNNs.

Chapter 4 explains key features of CNNs, DNNs, and PCA-based deep convolutional networks. It features an overview and motivations for the proposed method based on PCANet in object recognition and plant identification. It then discusses our contributions to existing state-of-the-art deep learning systems utilizing principal component analysis. The chapter concludes with a description of spatial pyramid pooling and classification methods employed in the final stage of this architecture.

Chapter 5 describes experiments conducted to evaluate our proposed system on the LifeCLEF 2014 plant identification datasets. It first represents our performed experiments to adjust optimum parameters for the proposed system, and then describes carefully designed experiments for testing variations in pose (translation, scaling, and rotations) and

illumination. The thesis concludes in Chapter 6 with the summary and discussion of obtained results.

Chapter 2

The Plant Identification Problem

In this chapter, we briefly review the main approaches of object recognition and learn about the motivations for performing plant identification. Next, we review the leaf recognition systems and common features used for leaf shape analysis including shape features, texture analysis, venation extractions, contour signatures, etc. Finally, we present the key highlights of top-ranking plant identification systems submitted to ImageCLEF 2012 and LifeCLEF 2014.

2.1 Object Recognition

Object recognition is defined as perception of familiar items in a digital image or video. In a more complete sense, it is defined as recognizing 3D objects from the scenes which, in the absence of depth sensors, are mapped as 2D images with different viewing conditions through optical sensors. Although humans use high-level visual perception skills to recognize familiar objects in real and digital settings, automatic object recognition relies on matching new items with previously learned information. In other words, automatic object recognition is largely built on concepts and algorithms from machine learning, pattern recognition, computer vision, and image processing.

Object recognition and detection are carried out in two different perspectives: specific (instance-level) and generic (category-level). Specific object recognition is the problem of matching a specific object or scene or identifying instances of a particular object. In this context—where concepts are based on matching and geometric verification tasks, local features are selected, detected, and extracted by, for example, automatic scale selection

and Harris and Hessian detectors. Scale-invariant feature transform (SIFT) [21] and scale-invariant region detection are two other methods applied in this category [22].

The category-level approach is the problem of recognizing the category of objects or scenes using, for instance, feature descriptors such as histograms of oriented gradients (HOG) [23]. In other words, the generic object recognition is concerned with recognizing various instances belonging to one category and classifying them into a similar class. It usually consists of statistical models of shapes or appearance learned from training examples. The most common approach for this categorization problem is collecting images from all the given categories, extracting features or patterns, and learning new models—usually a supervised one—which should make new predictions about existences or absence of objects in the new test images. This approach uses window-based and part-based models that acquire holistic descriptions or locally connected parts, respectively.

As we see, computer-based finite classification relies on objects' shapes, color, textures, and the like in a given illumination condition which make it very limiting and application specific; however, humans even consider functions of visualized objects while classifying them. The following section discusses plant identification as one of the challenging tasks in object recognition which attempts to match a specimen plant to a known taxon. More precisely, plant identification implies comparing certain characteristics and then assigning a particular plant to a known taxonomic group, ultimately arriving at a species or infraspecific name.

2.2 Plant Identification

Increasing our understanding of the earth flora is essential due to the role of plants in nutrition of humans and herbivorous animals, regulation of climate, and maintenance of land and soil structure against natural disasters such as floods and drought. For instance, recognizing and enlisting crops helps governmental organizations in setting agriculture policies for increasing productivity in harvesting crops as well as farmers in identifying diseased crop and determining the suitable herbicides and pesticides [24]. Food industry also needs to carefully determine the raw herbs and plants for manufacturing of their products. Furthermore, fields such as pharmacy and pharmacology continuously use herbs in medicines. The aforementioned applications are a few examples that show the need for

plant identification. Yet, modifications on the ecosystem which put more plant species into the threat of extinction [25] together with the shortage of skilled botanists and taxonomists have been driven forces in demands for automated plant identification systems.

Machine vision techniques combined with computer vision algorithms have long been used to locate and identify plant species [24]. These automatic plant identification systems in general use a database of digital images from known plant species and their organs as their knowledge base—one similar to that of the Royal Botanic Gardens. They are expected to provide the correct labels (species' names) and/or botanical information such as taxonomic information, place and date of collection, usual living locations, climate habits, etc. Some of these systems follow a set of questions about the plant morphology or taxonomic keys to narrow down the divisions and identify the sampled species from closely following criteria of taxonomy classification. These systems have a high level of interaction with users [26]. Other ones use probabilistic machine learning and computer vision techniques to provide rankings or votes for the possible categories to which the photographed species belong. These systems can be of use in hand-held devices and personal digital assistants (PDAs) to help farmers, engineers, and scientists in the fields.

Among various plant organs, leaves are the most commonly studied ones due to being more accessible than other organs. Moreover, leaves can be sampled year-round from evergreen perennials and relatively in shorter intervals from annual trees [25]. Besides leaves, shapes of flowers, fruits, and branching structures are decisive parameters for identification of not only species but also genera and plant families. Nonetheless, as alive and dried specimen can suffer from damages, deformations, diseases, and insects, automated identification and classification systems must be robust to such intra-class variations that affect the structural information.

Other problems with the plant images captured from the natural scenes include occlusion with other objects and a wide range of illumination changes in addition to varying viewpoints which increase the necessity of implementing complex plant identification systems able to learn as many features as possible. The most common features used in the literature and plant identification contexts are morphological features (MFs) including shape, color, texture, illumination, and geometrical features in addition to observation and photographer information. Among the morphological features, 2D outline shape of leaves and petals, leaf margin characteristics, and vein network (venation) structure are

the most useful features to which probabilistic computational techniques, machine learning, and pattern recognition have been applied. These features are of course low-level, while newer algorithms such as deep neural networks utilize high-level information, details of which will be explained in the following chapter.

It is worthwhile mentioning that although classification in computer science is defined as assigning a sample to one of the finite number of discrete categories [27], in taxonomy and botany it is the process of grouping individual samples based on their similarities to detect and define taxa, species, or genera [28]. However, our use of classification in this thesis is in line with the common definition in the field of computer science.

2.2.1 Image Acquisition and Preprocessing

Most plants have a variety of functional organs such as roots, stems, branches, leaves, flowers, fruits, and seeds whose shapes, sizes, and colors are largely varied. A thorough identification of plants from these organs requires full inspection of the specimen in the 3D form. However, as mentioned before, perceiving information related to 3D objects from their 2D images is a hard task for computers. In other words, eliminating depth from images make it difficult for artificial intelligence to correctly recognize the species to which those captured images belong. Still, among the aforementioned plant organs, leaf images are the easiest to identify and categorize as disregarding depth information in them affects the identification process considerably less than other organs.

In most plant species, leaves are grouped in clusters; hence, the majority of early efforts on automatic plant identification was concerned with acquisition, preprocessing, feature extraction, and supervised learning from isolated leaf images. Isolated leaves refer to single leaves that are plucked from their plants, cleaned, then either color scanned or photographed with a digital camera. The benefit of this method is that there is no background image or scene occluding the leaf and the 2D details of leaf structure will be clearly visible. Figure 2.1 shows a typical isolated leaf along with its main features and botanical terms.

In order to preprocess images of isolated leaves, one should consider the prospective features to be extracted from the image. As an example, [9] effectively uses shape and texture information for isolated leaves. Since scanned leaf images usually have shadows and uneven illumination conditions on uniform backgrounds, this proposed method readily

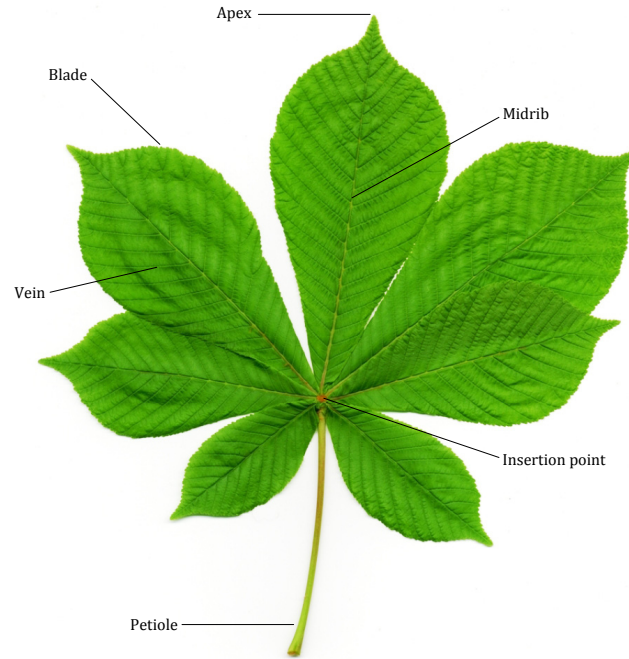


Figure 2.1: The main features and botanical terms of a typical leaf

segments leaves through edge preserving area attribute filters and adaptive thresholding. Next, it aligns major axis of leaves with the vertical axis and normalizes all heights to preserve the aspect ratio. After this size normalization, it uses PCA and leaf petiole's location to perform orientation normalization.

As mentioned before, color information is a key feature in most object recognition tasks; however, it is not highly discriminative in detecting leaf images as plant species usually have green shades and the variety of these shapes are affected by changes in the atmosphere, seasons, age, water, and nutrients [25, 29]. In addition, old and dried leaves of most annual plants become brown while completely or partially maintaining their edge and vein shapes. Therefore, RGB images are rarely used directly and the gray component of pixels is extracted from the RGB information. Accordingly, the region of interest (ROI) and the image contour can be extracted from the grayscale image.

2.2.2 Common Methods for Leaf Analysis

Besides general approaches proposed in object recognition literature such as histograms and shape matching, a number of methods frequently used for plants and especially leaf recognition utilize shape analysis including shape features, contour and landmark analysis, Fourier analysis, etc., texture analysis, and venation analysis. In this section, we will

review some prevalent methods proposed for leaf recognition.

2.2.2.1 Shape Analysis

Shape differences are more obvious in leaves than other features such as size, venation, or margin characteristics. In fact, shapes are determined by genetics while other features can be affected by environmental conditions. In this section, we mention prevalent approaches for leaf shape analysis.

Shape Features

Several publications have used leaf morphology and spatial parameters for plant species identification [24]. Morphological features are in fact statistical shape descriptors invariant to pose variations and are extracted from leaf contours as geometrical and invariant moment features [25, 29]. The following is a list of most commonly used quantitative morphological features used in the literature.

1. Aspect ratio: It is the ratio of the maximum length to the minimum length of the leaf's minimum bounding rectangle (MBR).
2. Rectangularity: It is the ratio of areas of the ROI to MBR.
3. Area ratio: It is the ratio of the ROI area to the convex hull.
4. Perimeter ratio: It is the ratio of perimeters of the ROI to the convex hull.
5. Sphericity: It is the ratio of radii of ROI incircle to excircle.
6. Circularity: It is the ratio of mean distance of all bounding points from the ROI center and the quadratic mean deviation of the mean distance.
7. Eccentricity: It is the ratio of the length of ROI's main inertia axis to minor inertia axis.
8. Form ratio: It is the ratio of the ROI area to its perimeter squared.
9. Elongatedness: It is the ratio of MBR length to its width.
10. Invariant moments: They are composed of seven invariant moments computed from central through the third order moments defined by Hu [30].



Figure 2.2: Leaf shape features. From left to right: convex hull, ellipse, MBR, and incircle and excircle. Adapted from [29]

11. Linearity: It is a parameter determined through the object's principal axis moment of inertia.

Figure 2.2 shows concepts of aforementioned shape features for a typical leaf ROI.

Usually, these region-based features are combined with k -nearest neighbors (k -NN) classifiers [7] or more efficiently, with moving median centers (MMCs) hypersphere classifiers [29, 31] to produce better results. MMC considers each pattern class as a group of hyperspheres and strives to have all points of a class covered by some hyperspheres and removing redundant hyperspheres encompassed by larger ones. However, the problem with the aforementioned quantitative measures is that not only they are not unique for a specific class of species with a large intra-class variation, but also they are highly correlated with each other. In other words, it is quite difficult to choose a set of sufficiently independent features that would describe and distinguish plant classes from each other.

Contour Signatures

A shape contour signature is a vector sequence of values calculated at the leaf's outline points in clockwise or counterclockwise directions. Signatures such as the centroid-contour distance, centroid-angle, and tangents to the outline can represent shapes independent of the leaf's location and orientation. To make the values independent of leaf size, one can perform normalization to the signatures. Methods of time-series analysis can then be applied to the calculated values to increase the system performance [32]. However, these boundary-based methods bring limitations in sections where two parts of the leaf intersect with each other. A proposed method was to remove darker areas from overlapping regions but it only worked where the acquired images were from thin or backlit leaves [33].

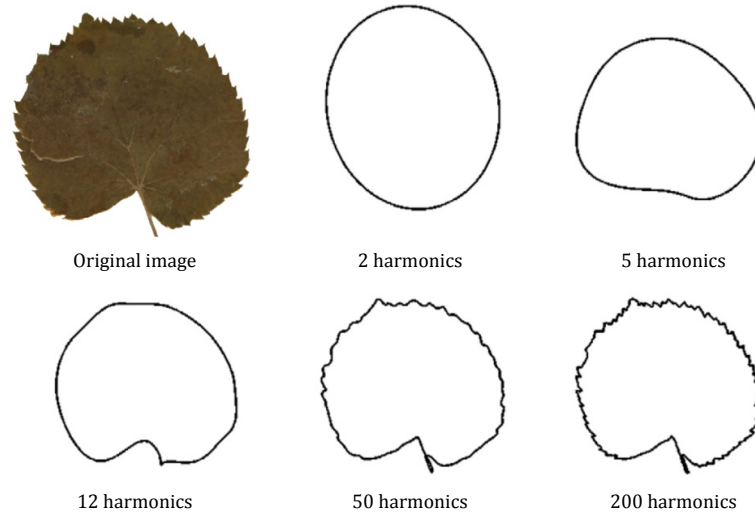


Figure 2.3: An example of EFA. Increasing the number of harmonics improves the precision and preserves more details. Adapted from [25]

Landmarks Analysis

Landmarks are points biologically definable and specific to certain species. They can be used to determine shapes of plant organisms by performing angular and linear measurements between the points. Landmark studies are mostly focused on certain species which clearly have the required features; this in turn requires strong knowledge about certain domains and families of plants. One can refer to leaflet features, spatial characteristics of lobes, and measurements of petioles as morphometrics used in the literature [25].

Elliptic Fourier Descriptors

Elliptic Fourier analysis (EFA) is a frequency domain analysis which calculates a set of Fourier harmonics or elliptic Fourier descriptors (EFDs) with only four coefficients from the outline [25]. Increasing the number of harmonics improves the precision of descriptors as displayed in Figure 2.3. Following this step, PCA is used to reduce the dimensionality. Also, normalizing EFDs enable them to represent leaf shapes independent of their sizes, locations, and orientations. In general, EFDs are useful for invariant moments and landmark measures [34]. A large variety of supervised learning schemes including artificial neural networks and support vector machines (SVMs) have been applied on these processed features for classification purposes [16, 35–37].

Fractal Dimensions

Fractal dimension is a measure of complexity of an object and is a real number that explains how completely a shape can fill its dimensional space. Several studies have used fractal dimensions for leaf identification [25]. Some of them utilized the multi-scale Minkowski fractal dimension or combined them with curve Fourier descriptors. [38] used the linear discriminant analysis (LDA) classifier on fractal information while [39] applied clustering techniques and obtained a 100% classification accuracy on a database with a few number of species. However, these features are not enough for a full description of complexity parameters and should be combined with other morphological features.

2.2.2.2 Texture Analysis

Besides analyzing leaves' shape, there are several algorithms applied on texture windows from digital leaf images to extract texture features. Among these schemes, one can mention multi-scale fractal dimensions, Gabor filters, wavelet transforms, Fourier descriptors, and grayscale co-occurrence matrices [25]. Still, these features are more informative when used together with outline-based shape analysis.

2.2.2.3 Venation Analysis

The pattern of veins or venation in leaves is quite conserved and unique within many species, and veins coarse structure can be used for leaf identification. Besides running smoothing and edge detection algorithms [40] and independent component analysis (ICA) on leaf images [41], researchers have developed classifiers for vein pixels from genetic algorithms [42]. Even though veins are largely studied after shape features, the results have not been very successful so far.

2.2.2.4 Segmentation

Besides morphometrics and general object recognition schemes, segmentation is also a very common approach used alongside other features for identification of plant images. There are many different methods applying interactive segmentation using shape context features, histogram-based features, morphological and geometric features, Markov random fields (MRFs), Gabor filtering and fractal dimensions [25]. On the other hand, vision

systems can be tested in detecting differences in radiation reflecting from leaves and soil surfaces and consequently segmenting leaves in images [24]. These algorithms were based on the knowledge that wavelengths in the range of 0.4 to 0.7 μm in the visible portion of spectrum have higher reflectance from soil than vegetation while the near-infrared region has more reflectance in green vegetation. Therefore, by changing the radiation illumination from near-infrared to visible spectrum and capturing images with charge-injection-device (CID) cameras, pixels focused on vegetation and soil surfaces will show variations in the spectral responsivity (amps/watt). To use this fact, the RGB layers are extracted from digital images and intensity gradients for each grayscale image are obtained. The leaf border will then exhibit the largest values of intensity gradients and be consequently used for segmentation and determination of leaf shapes.

2.2.3 Common Methods for Flower Analysis

Shifting our focus away from leaves, we can find studies that used morphometrics for flowers. Color is indeed a more discriminative feature in flowers and there have been methods using color-based segmentation with good results [43]. More successful results were obtained by combining angle code histograms and centroid contour distance to form a classifier and these methods showed that shape and outline information cannot be neglected for identification purposes [44].

As it can be seen, although the majority of studies on plant identification mentioned in the literature have dealt with leaves, a fully applicable and robust automatic plant identification system needs to have a database of all plant organs and be able to classify new samples and observations from a variety of illumination conditions, view points towards plant organs, image qualities, and the like. In recent years, CLEF campaigns such as ImageCLEF [10–12] and LifeCLEF [13, 14] have provided huge datasets with various categories to represent images from all the different organs of plant species including leaves, branches, stems, flowers, and fruits living in a particular geographic location—mostly France. CLEF is devoted to promoting and evaluating multilingual and multi-modal information retrieval systems and the main goal of these competitions is to benchmark the challenging task of content-based identification and retrieval of plant species. Therefore, teams participating in their annual plant identification challenges strive to pro-

vide accurate and robust systems from all of the provided categories. In the following section, we review the most prominent, fully automatic systems participating in plant identification tasks of the ImageCLEF 2012 and LifeCLEF 2014.

2.2.4 Highlights of Plant Identification Systems in CLEF Campaigns

Most of the early plant identification systems proposed for CLEF campaigns build on the rich literature for leaf identification, and use either one type of shape, color, and feature textures or a combination of them as reviewed in Section 2.2.2 [31, 45, 46]. The following includes a brief review of the highest scored teams in ImageCLEF 2012 and LifeCLEF 2014 which were concerned with leaf recognition and plant identification from scanned images, scan-like photographs, and unconstrained photographs of different organs of plants.

The ImageCLEF campaign started in 2011 with an image-based task covering over 73 plant species and went on to include 126 plant species in 2012. As one of the participants in ImageCLEF 2012, INRIA Imedia PlantNet [47] combined boundary shape information and local features as a complex shape context descriptors. They used automatic segmentation to extract shape features and decrease the effect of photograph background by extracting local features around Harris points.

The LSIS/DYNI group did not use any segmentation for their submission to the photo category, but performed feature extraction with spatial pyramid pooling (SPP) [48]. For the large-scale classification, they used linear SVM to select the estimated assignments based on the one-vs-all multi-class strategy. They also submitted a run with sparse coding of patches, dense SIFT features, as well as dense multi-scale color improved local binary patterns (LBPs).

The Sabanci-Okan system for ImageCLEF 2012 [16] used shape, color, and texture features as well as quasi-flat zone-based color image simplification combined with powerful classifiers. For photographs from natural backgrounds, they assumed that the leaves of interest occupy the center of the photograph and that they have a single dominant color [9]. Of course, this feature extraction makes it a challenge to add information about the natural setting of plants unless the background and foreground are separated. Next, they performed a morphology-based image partitioning method to create flat zones based on local and global spectral variations. This aggressive segmentation left only one leaf

in the image center and reduced the problem to an isolated leaf recognition system. Although, it would be used alongside the local invariants approach, it eliminated a lot of useful resources about the image background and other visual information [37].

When it came to photographs in the natural background category in ImageCLEF 2013 including 250 species, Sabanci-Okan system [37] used texture features for classification of stems, texture and shape for leaves, and texture and color information for fruits, flowers, and entire plant groups. In the stem category, it calculated the maxima and horizontal and vertical derivatives to determine the orientation of stems, next cropped two thirds of the image surface to centralize the image and remove the background information. The IBM Australia performed the same cropping in LifeCLEF 2014 [49].

Later in the LifeCLEF 2014 campaign, the database size had increased to over 500 species and besides the image-based plant identification task, an observation-based task was added based on several detailed pictures from different views of various organs of similar plants. These viewpoints or categories contained leaves, flowers, fruits, stems, branches, entire view, and scanned leaves. The Pl@ntNet team [50] treated each organ/view independently, extracted the visual content from the lowest local levels in pictures and applied a hierarchical fusion framework to combine these visual contents with those in the highest levels. For preprocessing, they applied a rhomboid-shaped mask to each image and a Gaussian-like distribution to bring more points to the picture center. Depending on the picture visual content, between 150 to 200 local features were extracted in patches around those approximately 100 points. For the scanned leaves category, they used speeded-up robust features (SURF), edge orientation histogram (EOH), a 20-dimension Fourier histogram, and a 16-dimension Hough transform-based histogram. A series of hashing and local similarity search were used, and the number of matches between any training image and query image were calculated through lists of 30-NNs of each local feature. To improve the global performance for scanned leaves and scan-like photographs, automatic leaf boundary detectors were run to describe the leaf margin. Moreover, six morphological features including circularity, sphericity, convexity, rectangularity, solidity, and ellipse variance were extracted. Finally, in each of the four submitted runs, they used different fusion methods to combine several responses (from local and global features) for each training image.

The Sabanci-Okan team concentrated on the scanned leaves category in LifeCLEF

2014 and applied automatic segmentation through edge preserving algorithms using area attribute filters and adaptive thresholding. Through these algorithms, they extracted various morphological and texture features similar to [9]. These features were used for stem category classification as well. Their submissions also contained identification of flowers, fruits, and entire categories for which they used a bag-of-words (BoW) model and dense-SIFT feature extraction followed by k-means clustering. To compute scores for prediction of species belonging to each class, a SVM classifier was used. They also implemented an 8-layer CNN for score prediction in the branch and leaf categories.

In the same campaign, the BME TMIT team used dense SIFT for feature detection and description followed by the PCA [51]. They next applied a BoW model to complete the high-level image descriptors by calculating a Gaussian mixture model (GMM)-based Fisher vector to determine high-level image descriptors. Finally, they utilized a C-support vector classifier with the radial basis function (RBF) as the kernel. However, they obtained their best results through a combined classifier with the weighted average of classification reliability values at each viewpoint or category.

IBM Australia team [49] achieved the highest inverse rank scores in LifeCLEF 2014 plant identification task. They implemented an efficient GPU-based deep CNN with five convolutional layers, some of them followed by max-pooling layers, and three fully-connected layers as well as a final softmax layer. After automatically specifying the region of interest in each image, their first submitted run included multiple low-level extracted features. These complementary features were encoded with a Fisher vector to perform accurate linear classification. Besides having a complicated and efficient deep CNN, their system utilized the image data with the annotation-provided metadata, and used classifier fusion. In their second run, they used Fisher kernel encoding, and extracted SIFT and color moments as dense features from raw images. Each feature was modeled with a GMM, turned into the Fisher vector representation, and used for training an individual linear SVM classifier. For the segmentation in the fourth run, their approach for the flower and fruit categories was to compare the pixel values of red and green channels and extract the more red zone as the region of interest. Finally, for the scanned leaf category, they normalized the background with a white color.

Chapter 3

Convolutional Neural Networks

This chapter covers the main architecture, properties, and advantages of a variety of deep learning methods built upon on the artificial neural networks. The structure of fully-connected feedforward neural networks, the neuronal units, weights and parameters, activation functions, learning process, and backpropagation algorithm are presented in detail. The discussion then continues with a thorough description of convolutional neural networks. The concepts and motivations for these structures are presented and the building blocks of these networks such as the convolutional, pooling, and normalization layers are then described and visualized.

3.1 Deep Learning

Visual content usually experiences large intra-class variability due to diverse lightings, non-rigid deformations, occlusion, and misalignment conditions. This variability makes image classification from visual content very difficult. The earlier efforts to tackle the intra-class variability used to fuse expert domain knowledge into pattern recognition or machine learning systems. More precisely, they would transform image pixel values or other raw data through a carefully designed low-level feature extractor into an appropriate internal representation. The obtained feature vector or representation would then be fed into a classifier or learner that would analyze the input patterns.

A number of these feature extractors have been introduced in the previous chapter; SIFT and HOG are among the famous ones for object recognition. Moreover, local binary patterns and Gabor filters are mostly used for face and texture classification. These low-

level feature extractors have been successful because they were especially designed and tailored for their specific fields, data structures, and required tasks. It follows that, any new feature extractor requires having updated expert domain knowledge before it can be adapted to new problems.

However, in contrast to traditional machine learning methods where the features are chosen manually and extracted through instructed algorithms, representation learning is composed of methods that first feed pixel values or raw data into the system and allow it to automatically discover features or representations that can be used for detecting, distinguishing, and classifying patterns [17]. Deep learning methods utilize representation learning in multiple levels; the raw data is fed as the first representation and, at every level, nonlinear modules transform the representation to more abstract level. Therefore, a deep neural network learns abstract representations so that they bring more invariance to the problem of intra-class variability.

In a deep convolutional network, simple modules are stacked in a multi-layer structure. All or most of these modules can learn the representations and a lot of them indeed compute nonlinear mappings between the input and output. The transformation of inputs within each module increases the representations' selectivity and invariance. As the number of nonlinear (transformation) layers or representations increases from, for example, 5 to 20, the deep network becomes capable of implementing very complex functions. These functions are generally sensitive to fine details in the inputs but insensitive to large variations such as lighting, pose, background, and surrounding objects.

In consequence, higher-level features of natural signals in DNNs are composed of modifications of lower levels. Suppose the input data to the network is the array of pixel values from an image. There might be edges in the image at specific locations or orientations, and the first layer learns a representation which shows presence or absence of these edges. These edges might have particular arrangements or local combinations, and the second layer typically discovers these motifs in the image. These motifs could have been assembled as larger, more familiar compositions, and the third layer then detects these objects. Likewise, higher layers detect more complex assemblies important for pattern discriminations and ignore or suppress variations irrelevant to classification.

As can be seen, the main difference between deep learning and traditional machine learning is that feature layers are not designed or imposed by humans; instead, the net-

work learns those features from data through a general-purpose learning procedure. The learning procedure itself can be done through either supervised or unsupervised learning. However, in unsupervised learning, the system can benefit from the automatic process in which no information about labels of the output is used alongside the learning system. Therefore, deep learning is very useful in learning and discovering fine and complex structures in high-dimensional data—a problem which artificial intelligence and pattern recognition have failed to solve thus far.

3.2 Artificial Neural Networks

Artificial neural networks belong to the statistical learning models and are used to process information in machine learning problems. Inspired by the central nervous system (CNS) in humans and animals, they are composed of processing units or neurons (artificial nodes) which have weighted interconnections with each other throughout the system and approximate nonlinear functions of their input signals.

Conventional computers use cognitive or algorithmic methods to solve problems that we understand and for which we already have an algorithm and solution; but neural networks learn by examples and process information similar to the human brain. These examples should be selected carefully so that the learned algorithm is not specific to a problem and does not cause overfitting for test instances.

3.2.1 Historical Background

In 1943, McCulloch and Pitts [52] developed models for neural networks based on mathematics and algorithms known as threshold logic. Those models used several assumptions about simple functional approximations of neurons and were the basic simulating biological processes for artificial intelligence. Later, based on the mechanism of neural plasticity, Hebb proposed a hypothesis known as Hebbian learning for unsupervised learning rules [53]. In the mid 1950's Farley and Clark [54] and Rochester *et al.* [55] used pioneer computational machines to simulate Hebbian networks.

Perceptron was created later by Rosenblatt [56] as a pattern recognition algorithm formed by a two-layer learning network that used only addition and subtraction. At the end of 1960s, however, Minsky and Papert showed that single-layer neural networks

could not process and simulate XOR circuits [57]. Since larger networks required longer processing time and available computers of that era did not have such processing powers, the neural networks research declined for a while. It was in 1972 when Klopff [58] proposed a learning method for artificial neurons based on heterostasis, a biological principle for learning of neurons. Three years later, Werbos [59] developed the backpropagation learning method for networks of perceptrons which have multiple layers, use different threshold functions in the neurons, and utilize more robust learning rules. Using these new learning methods, artificial neurons now could solve the XOR problem with one hidden layer. The one-hidden layer perceptrons can use hard-limiting functions to build any unbounded convex region.

The growth of neural networks declined gradually as support vector machines and linear classifiers became more popular in machine learning techniques; however, deep learning concept and architecture renewed a global interest in neural networks since the end of 2000s.

3.2.2 Model of Biological Neuron

The human brain is similar to a highly complex and nonlinear computer and is composed of about 10^{11} neurons with around 10,000 connections per neuron. A typical neuron has a series of fine structures or receptors called dendrites which receive signals from other neurons through a connection called synapse. The electrical activity transfers throughout axon, the longest part of neuron usually covered with a thin insulating layer called myelin. The transfer of electrical activity is done through movement of ions that trigger spikes along the axon. The final part of each axon is split into thousands of branches which once again are close to the dendrites of the next neuron, and the electrical activity from the first axon excites chemical substances whose motion to the next dendrites is equal to inhibition or excitation of electrical activity throughout the synapses of connected neurons. When a neuron receives an excitatory input sufficiently larger than its inhibitory one, it sends a spike along its axon. Figure 3.1 shows the structure of a biological neuron.

In the human brain, learning is done through adjustments of synaptic connections, i.e. changes to the effectiveness of synapses such as forming new synaptic connections or detaching of some other ones. The brain has another property called neuroplasticity; i.e., although the brain at birth has a series of shaped networks from interconnected neurons,

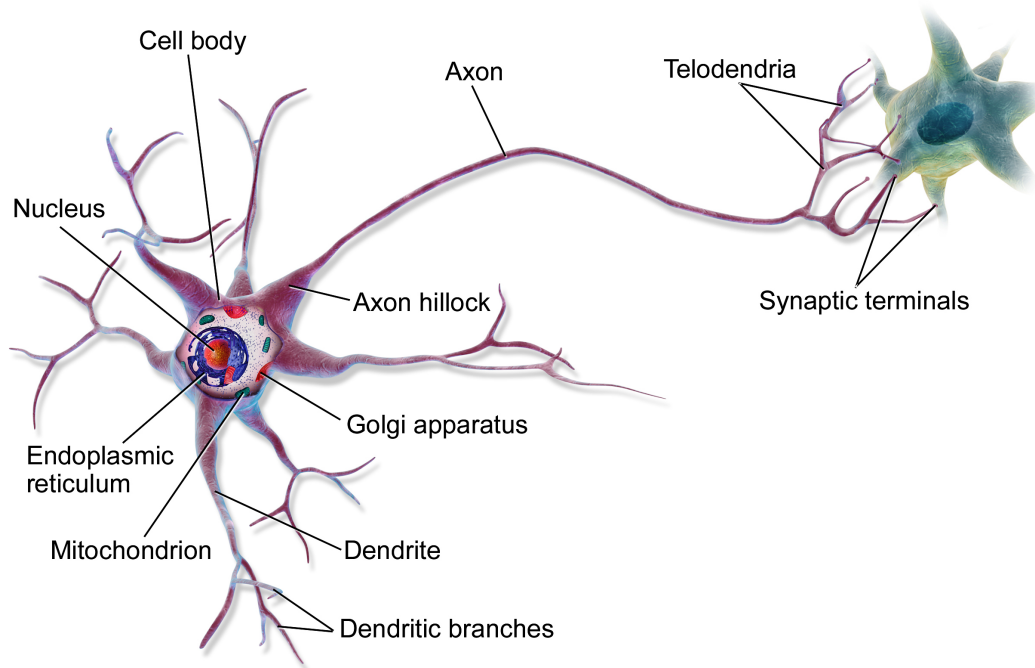


Figure 3.1: The structure of a biological neuron showing its synapse with a neighboring neuron. Adapted from [60]

new connections are built in response to new inputs and adaptation to the environment.

3.2.3 Perceptron

A perceptron is a common type of single artificial neuron that computes the weighted input of one or more binary inputs and uses a threshold activation function to produce a single binary output. Figure 3.2 represents the model of a perceptron (artificial neuron) which receives electrical activity from other neurons and applies a hard-limiting activation function on the weighted summation of the activities.

Rosenblatt introduced real-valued weights, \mathbf{w} , to emphasize the importance of their respective inputs, \mathbf{x} , in the calculated output. The weighted sum $\mathbf{w}^T \cdot \mathbf{x} = \sum_j w_j x_j$ plus the value of bias, b_i for the i -th neuron, is then compared with the threshold values of the activation function and its output becomes 0 or 1, depending on the result of comparison. In principle, the perceptron separates the input space to two regions divided by the hyperplane $\mathbf{w}^T \cdot \mathbf{x} + b_i = 0$. However, a single-layer perceptron has a limitation in that it can only learn linearly separable problems. For linearly not-separable problems, a usual solution is to use multi-layer networks and the backpropagation algorithm both discussed in the following sections.

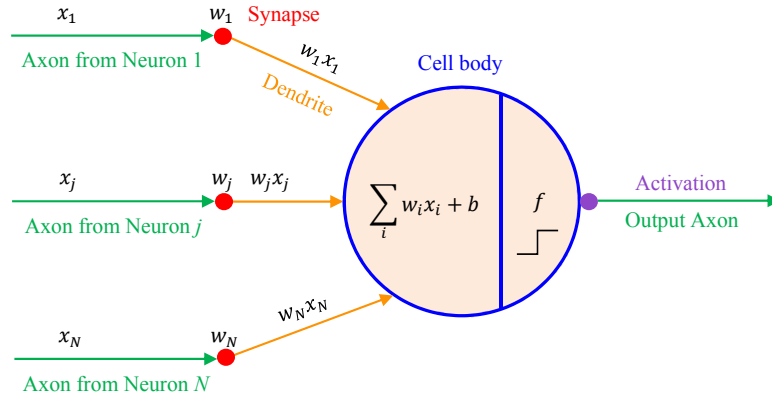


Figure 3.2: Model of a perceptron

Obviously, the weights, types of activation function, and the threshold values are parameters of the perceptron. Modifying these parameters changes the decision-making criteria and enables us to implement different binary functions and solutions. In addition, similar to the mechanisms in biological neurons, connections could be excitatory or inhibitory. Here, positive weights show excitatory connections while negative weights represent inhibitory connections.

3.2.4 Activation Function

The weights and mappings between the input and output of an ANN unit determine its behavior. To be specific, if the function represents the activation function of the i -th neuron, the output is found from the following equation

$$a_i = f(n_i) = f\left(\sum_j w_j x_j + b_i\right) \quad (3.1)$$

where n_i is the net input plus the bias term.

The activation function can take various shapes but usually it is in the form of hard-limiting, log-sigmoid, linear, ramp, etc. For the hard-limiting function,

$$a_i = \begin{cases} 0, & \text{if } n_i < 0 \\ 1, & \text{if } n_i \geq 0 \end{cases} \quad (3.2)$$

For the log-sigmoid function, $a_i = \frac{1}{1+e^{-n_i}}$ and in the linear units, $a_i = n_i$. The former gives the most precise approximation to the behavior of real neurons. The type of acti-

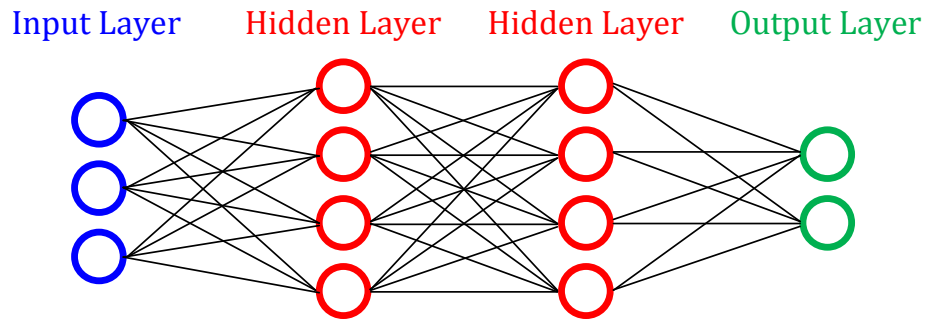


Figure 3.3: A fully-connected three-layer neural network

vation functions used within each layer is usually consistent. Generally, the log-sigmoid activation function is used in the hidden units. In the classification problems, the sigmoid or linear activation functions could be used while the approximation/regression problems usually use linear functions at the output neurons.

Currently, the most popular nonlinear function in the CNNs is the rectified linear unit (ReLU) or the half-wave rectifier. The output of this activation function is

$$a_i = \max(0, n_i) \quad (3.3)$$

A smoother approximation to the ReLU is $\ln(1 + e^{n_i})$. Until the introduction of this function, neural nets used smoother nonlinearities such as $\tanh(n_i)$ or log-sigmoid. But ReLU typically has a much faster learning rate in networks with a lot of layers.

3.2.5 Neural Networks Architecture

The prerequisites for building a neural network for any specific task are designing the connections between the units and imposing right weights on them to determine the strength of influence between two units. These connections simulate the synaptic connections between neurological neurons used for storing the acquired knowledge.

The most common type of artificial neural networks has one layer of input, two hidden layers and one layer of the output unit. Figure 3.3 shows a fully-connected three-layer neural network.

Besides the input unit which receives the raw information into the network, activities of hidden units are functions of the input unit activities and the input-hidden unit connection weights. Likewise, the output's behavior is determined by activities of hidden

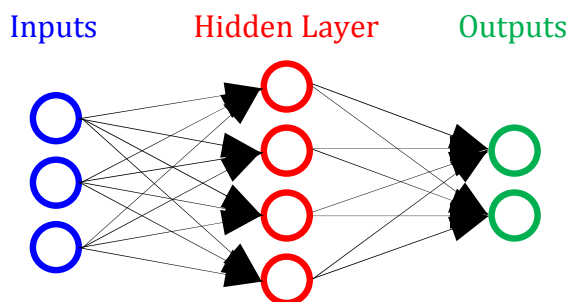


Figure 3.4: A two-layer fully-connected feedforward neural network

units and weights of connections between the hidden and output units. The hidden units essentially modify the input data in a nonlinear way enabling the last layer to make the categories linearly separable.

The active condition of each hidden unit is determined by weights of connections between the input and that particular hidden unit; therefore, changing these weights enables the hidden units to choose their own representations. As a result, multi-layer neural networks are interesting as the hidden units are almost free in constructing their own representations from the input data.

3.2.5.1 Feedforward Networks

The feedforward ANNs are straightforward networks that associate inputs to the outputs in one direction. As there is no loop among the connections, no output in any layer affects the same layer. The feedforward networks are also known as bottom-up or top-down architectures and an example of them is shown in Figure 3.4.

The feedforward neural network architectures are used in many applications of deep learning. For example, in solving category-level object recognition problems, they receive input images as fixed-size inputs, and map them to fixed-size outputs such as the probability of belonging to each of the many categories.

3.2.5.2 Feedback Networks

In feedback (recurrent) networks, some of the inputs are connected to some of the outputs. In other words, some of the signals can travel in both directions due to the existence of loops in the network. Feedback networks are dynamic and their state changes continuously until an equilibrium point is reached. They remain at this point until the input

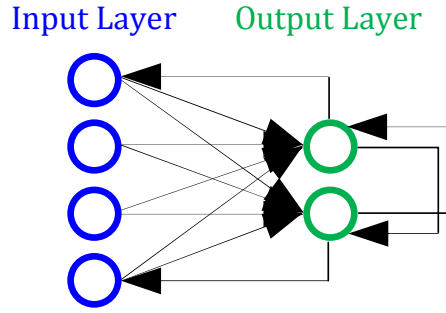


Figure 3.5: An example of feedback (recurrent) neural network

changes which requires finding a new equilibrium point. When located in single-layer neural networks, feedback architectures are called recurrent structures. In larger networks with at least one hidden layer, however, they are referred to as interactive networks. Figure 3.5 shows an example of feedback neural network.

3.2.6 Learning Process

For a neural network to learn a particular task, a number of steps need to be performed. First, a set of training examples will be presented to the network showing the pattern of input activities and the desired activities (or labels for the output units). Suppose the i -th perceptron of a single-layer neural network has an activation function calculating a numerical output label $a_i = \text{hardlim}(\mathbf{w}_i \cdot \mathbf{x}_j)$, where \mathbf{w}_i is the row vector of connection weights for the i -th unit and \mathbf{x}_j is the column vector of the j -th input instance. By comparing this calculated label and the actual label y_i , we determine how closely these two output labels match each other and change the connection weights so that the network can produce a better approximation for the desired labels.

The weights are updated using the following formula known as the tentative learning rule

$$\mathbf{w}_i^{\text{new}} = \mathbf{w}_i^{\text{old}} + (y_i - a_i)\mathbf{x}_j \quad (3.4)$$

where $(y_i - a_i)$ equals the i -th error term e_i . If the perceptron has a bias term such as b_i , it will also be updated using the following formula

$$b_i^{\text{new}} = b_i^{\text{old}} + (y_i - a_i) \quad (3.5)$$

This learning rule can be easily generalized to larger, multi-layer neural networks. Also, this learning rule represents modification of the information stored in the network as a function of the experience. Nevertheless, not all networks are flexible and able to update their weights. In such fixed networks, $dW/dt = 0$ and weights are fixed *a priori* according to the problem to be solved. On the other hand, the networks that we just reviewed are adaptive and able to change their weights, i.e. $dW/dt \neq 0$.

3.2.6.1 Supervised Learning

One of the most common paradigms in machine learning is supervised learning. In this case, through the help of an external teacher, a training set composed of input-output pairs (X, Y) is given to the system. In other words, each output units knows what desired output value or label it should produce in response to input signals. Supervised learning is then defined as learning an unknown function f such that $f(X) = Y$ where X is an input example and Y is the desired output.

The main tasks in supervised learning are classification, concept learning, and regression. Classification is the main theme of this thesis and is defined as assigning an object/event to one of a given finite set of classes or categories. In supervised learning from a database of images, each image is labeled with its category. During training, each image is shown to the machine which produces a vector of scores, one element for the probability of belonging to each category. The desired or correct category should be the one with the highest score among all categories, so during training, an objective function is computed for measuring the distance between the output scores and desired pattern of scores. The machine will then adjust its internal, adjustable real-valued parameters, often weights, to minimize this distance or error. Major paradigms of interest within the field of supervised learning are error-correction learning, stochastic learning, and reinforcement learning where a feedback in the form of positive or negative reward is given to the machine at the end of a sequence of steps.

To adjust the weight vector, the learning algorithm computes a gradient vector. This vector shows that how much the error would change and in which direction if the weights are increased by small amounts. The weight vector would then be adjusted to move in the opposite direction of the maximum gradient, also known as the steepest descent. As the number of features increases, so do the number of input units and the number of

parameters in the feature space. The goal is to minimize the objective function, averaged over all training instances, in the high-dimensional weight space.

A key issue in supervised learning is error convergence, i.e. minimization of the error term between the actual and computed outputs (labels). In this case, the problem becomes choosing the set of weights so as to minimize the error term, usually through mean square convergence. Another highly practical method is the stochastic gradient descent (SGD). In this method, every time a small set of input pairs is given to the system, outputs, errors, and the average gradient are computed and weights are adjusted accordingly. This process repeats for several small sets of input pairs from the training set until the average of the objective function stops decreasing. As each small set of input pairs creates a noisy estimate of the average gradients, this process is named stochastic gradient descent.

3.2.6.2 Unsupervised Learning

Unsupervised learning is another paradigm of machine learning in which no desired output or external teacher is used alongside the training system. This paradigm self-organizes the data fed to the machine and detects the collectively emergent properties. Hebbian theory and competitive learning are two paradigms within the unsupervised learning, and Fuzzy clustering and k-means are two common clustering methods used in this field.

3.2.6.3 Backpropagation Algorithm

One of the main motivations behind developing various pattern recognition techniques has been replacing manually extracted features with trainable multi-layer networks. It was around mid-1980s when researchers realized that they could train multi-layer architectures using simple SGD. Several different groups in 1970s and 1980s [59] independently discovered the idea that gradients could be computed using the backpropagation procedure assuming the output modules were quite smooth functions of their internal weights and input signals.

The backpropagation procedure computes the gradient of an objective function with respect to weights of its multi-layer stack using the chain rule for derivatives. Assume that the value of the j -th output unit O_j is a function (f) of the weights from units in the

previous layer (O_i 's) and the respective connection weights:

$$O_j = f(net_j) = f\left(\sum_i W_{ij}O_i\right) \quad (3.6)$$

The error derivative of the weights, defined as the partial derivative of error on the input pattern p with respect to each of its preceding connection weights in the previous layer, is calculated using the chain rule and the known partial derivative function as follows

$$\frac{dE_p}{dW_{ij}} = \frac{dE_p}{dO_j} \times \frac{dO_j}{dnet_j} \times \frac{dnet_j}{dW_{ij}} \quad (3.7)$$

Therefore, using 3.6 we will obtain

$$\frac{dE_p}{dW_{ij}} = \frac{dE_p}{dO_j} \times f'(net_j) \times O_i \quad (3.8)$$

where $\frac{dE_p}{dO_j}$ is the rate at which the error changes with respect to the activity level of the output unit. This is the general formula for the error derivative of weights for both linear and nonlinear units and any output or hidden-layer node.

However, if node O_j is an output layer node and Y_p is the true label of the input pattern p , using the definitions for the squared-error term we can simplify the partial derivative of error term as follows

$$E_p = (Y_p - O_p)^2 \quad (3.9)$$

$$\frac{dE_p}{dW_{ij}} = -(Y_p - O_p) \times f'(net_j) \times O_i \quad (3.10)$$

The preceding backpropagation equations may be applied repeatedly to propagate gradients backwards through all modules, i.e. start from the output layer producing the predicted values and go back to the external input layer. After computing these gradients, one can easily calculate the gradients with respect to other modules' weights.

3.3 Convolutional Neural Networks

Convolutional neural networks (CNNs/ConvNets) are highly similar to ordinary neural networks in that they are composed of neurons that have learnable weights and bias terms. Each neuron receives a number of inputs, performs a dot product of the weight vector and input vector, and optionally performs a nonlinear function on the resulted net input. CNNs have a loss function that could be implemented by SVM or Softmax on the last fully-connected layer. Moreover, all the concepts and methods developed and explained for learning in regular neural networks are still applicable to CNNs. However, CNNs take advantage of the fact that the inputs are images and not long vectors. In this way, CNNs can apply certain related properties into their architecture. Utilizing these properties helps with a more efficient implementation of forward functions and highly decreases the number of network parameters.

CNNs have been successfully used for detection, segmentation, and object and region recognition in images since early 2000s. Most of these tasks had labels and were used in fields such as brain and nervous system connectomics [61], segmentation of biological images [62], traffic sign recognition [63], as well as face, text, human body, and pose detection in natural images [17]. Until the launch of ImageNet competition in 2012, CNNs had been ignored by researchers in machine learning and computer vision fields. It was then that, after applying CNNs to a dataset with 1,000,000 images and 1,000 different classes, these communities noticed the magnificently low-error rates of CNN-based approaches. The success of CNNs lies in techniques for generating more training instances by deforming the available ones as well as the efficient use of ReLUs, GPUs, and dropout—a new regularization technique [64].

3.3.1 CNN Architecture

As discussed earlier, fully-connected ANNs receive an input as a single vector and transfer it to the consequent hidden layers composed of neurons. Neurons inside a single layer do not share any connections with each other but each neuron is fully connected to all neurons in the previous layer. Finally, the output layer sends out outputs representing the class scores in classification problems.

However, in CNN the neuronal units are not necessarily fully connected. Moreover,

each layer has a number of feature maps. Stacking convolutional layers followed by pooling layers constitute the first few stages of CNNs. Input layer includes the raw pixel values of images and units in each layer are connected to a local patch in the previous layer. Since the input image could be in the grayscale or color format (volume), the arrangement of values from the output neurons can be in the 2D or 3D space. The connection weights between these units and the preceding patches are called filter banks. The local weighted sums are then mapped to the next layer by passing through a nonlinearity mapping such as ReLU.

The role of convolutional layers is to detect local conjunctions of features from the previous layer while the pooling layers semantically merge similar features into one feature. Each pooling map takes the maximum of a local patch of neuronal units. In general, a full CNN architecture is composed of stacked input layer, convolutional layers, pooling layers, and the fully-connected output layer.

The inspirations for the convolutional and pooling layers of the CNNs have come from classic concepts of simple and complex cells in visual neuroscience [65]. The basis of CNN was partly based on Neocognitron [66]; nevertheless, Neocognitron did not use backpropagation or any similar end-to-end algorithm for supervised learning.

3.3.1.1 Convolutional Layer

The core building block of a CNN is the convolutional layer. In the convolutional layer, neurons are connected to local regions in the input and their outputs are dot products of the connection weights and the regional inputs. The convolutional layer, in summary, will compute the output of those outer neurons.

A set of filters with learning capability constitute parameters of the convolutional layers. In the forward pass, each filter is slid or convolved across the input image height and width and produces a 2D activation map of that filter. This is similar to calculating the dot product between the input and filter entries. In this way, the network will learn which filters activate seeing a specific pattern of features at some locations in the input. A full output map or volume is built by stacking all of these activation maps for all filters along image color maps. In contrast to the different feature maps in a layer that uses different filter banks, all units in a feature map share the same filter banks.

In CNNs with high-dimensional inputs such as images, it is not practical to make

the network fully-connected; therefore, the connections along the width and height are local as each neuron will be connected to just a local region (receptive field) of the input volume. In other words, every entry in the output volume is an output of a neuron looking at just a small region in the input and sharing parameters with neurons from the same activation map. However, connections will be full along the color maps.

Besides the number of color components which control the number of neurons, overlapping stride of receptive fields and zero-padding of input maps during convolution specify the spatial size of the output volume. Assume W , F , S , and P are the input map size, receptive field size of convolutional layer neurons, filtering stride, and the amount of zero-padding used on the input map borders, respectively. Therefore, the number of neurons fitting in the output map under the zero-padding condition is given by

$$n = \frac{W - F - 2P}{S + 1} \quad (3.11)$$

If the computed n is not an integer, then the strides are set incorrectly. Also, for $P = (F - 1)/2$ and $S = 1$, the input and output volume will have the same spatial size.

3.3.1.2 Pooling Layer

CNNs usually insert pooling layers after a number of successive convolutional layers. These layers perform downsampling operations along the width and height dimensions and gradually reduce the number of parameters and the required computations in the network. This in turn controls and reduces overfitting during training.

Subsampling by Max-Pooling

Max-pooling is a type of nonlinear downsampling that finds the most responsive node or unit with higher activation from a given region of interest. It partitions the input image into a set of non-overlapping rectangles; next, it outputs the maximum value for each of the sub-regions. Although there exist other different approaches for pooling such as average pooling, L2-norm pooling, stochastic and weighted pooling [67], biological studies have shown that human brain likely utilizes a max-pooling neural activation structure.

Max-pooling is a useful technique in vision. It reduces the required computation for further processing by eliminating non-maximal values. On the other hand, it provides a

form of translation invariance or robustness to position. Hence, max-pooling is a good approach for reducing the dimensionality of intermediate representations [68].

3.3.1.3 Normalization Layer

A CNN may also contain an extra layer for normalization purposes. As a local normalization scheme, response normalization was first introduced in [69] to implement inhibition schemes observed in the brain. When a neuron fires at a very high activation level, a local response normalization layer suppresses the activation in the surrounding neurons. This layer is defined based on three parameters α , β , and k and a convolutional structure or neighborhood shape. Activation level of a neuron in the destination (normalization layer) corresponding to the neuron x in the source layer is given by

$$\frac{f(x)}{\left(k + \frac{\alpha}{|N_x|} \sum_{z \in N_x} (f(z))^2\right)^\beta} \quad (3.12)$$

where $f(x)$ is the activation level of neuron x and N_x is the kernel or the set that contains the neurons in the neighborhood of x .

3.3.1.4 Output Layer

The output or final layer is generally designed to assign the obtained representations for an image to a specific class. The fully-connected and Softmax layers are two types of output layers used in CNNs.

Fully-Connected Layer

The highest level of representation in neural networks is implemented via fully-connected layers that proceed several convolutional and max-pooling layers. A fully-connected layer receives activations of all neurons in the previous layer and connects them to every single neuron in itself. Thus, the final activation can be implemented using a matrix multiplication summed with a bias offset, if needed. These layers are considered one-dimensional. They compute the class scores in these classification problems; thus no convolutional layers come after them.

Softmax Regression

Softmax regression or normalized exponential is a generalization of logistic regression to multi-class classification. It is an appropriate model for classification especially where there are no overlaps between different categories—the so-called mutually exclusive classes. On the other hand, if categories are not mutually exclusive, one can build and train binary logistic regression classifiers as the same one-vs-all technique [27].

Consider a set of sample-labeled pairs (x_i, y_i) , $i = 1, 2, \dots, m$, with feature vectors $x_i \in \mathbb{R}^n$. In the case of binary classification where $y_i \in \{-1, +1\}$, the logistic regression makes a decision based on the following hypothesis function

$$h_{\theta}(\mathbf{x}_i) = \frac{1}{1 + \exp(-\theta^T \mathbf{x}_i)} \quad (3.13)$$

where θ represents the model parameters including weights augmented with the bias learned from the training data by minimizing the following cost function

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \log(h_{\theta}(\mathbf{x}_i)) + (1 - y_i) \log(1 - h_{\theta}(\mathbf{x}_i)) \quad (3.14)$$

However, in multi-class classification problems, the goal is to estimate the conditional probability for each class label using the following formula

$$p(y_i = j | \mathbf{x}_i; \theta_j) = \frac{\exp(\theta_j^T \mathbf{x}_i)}{\sum_{j=1}^k \exp(\theta_j^T \mathbf{x}_i)} \quad (3.15)$$

where $p(y_i = j | \mathbf{x}_i)$ is the conditional probability of the j -th class label given instance vector \mathbf{x}_i , $j = 1, 2, \dots, k$. The denominator normalizes the output within $[0, 1]$. Consequently, the output vector of the Softmax regression hypothesis for a given test instance is

$$h_{\theta}(\mathbf{x}_i) = [p(y_i = 1 | \mathbf{x}_i; \theta_1), p(y_i = 2 | \mathbf{x}_i; \theta_2), \dots, p(y_i = k | \mathbf{x}_i; \theta_k)] \quad (3.16)$$

where the model parameters can be obtained by minimizing the following cost function

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m \sum_{j=1}^k \delta(y_i = j) \log(p(y_i = j | \mathbf{x}_i; \theta_j)) + \frac{\lambda}{2} \sum_{i=1}^k \sum_{j=1}^n \theta_{ij}^2 \quad (3.17)$$

where $\delta(\cdot)$ is the Dirac delta function and is equal to 1 only if its argument holds. Also,

the second term of this cost function with the regularization parameter $\lambda > 0$ is called the weight decay term and ensures the optimization problem is strictly convex with a unique solution. Moreover, since the minimization approaches apply the gradient descent or other iterative methods, the Hessian matrix will become invertible; hence, it will guarantee that the minimization algorithm for solving unconstrained, nonlinear problems converges to a global minimum.

Chapter 4

Object Recognition Using Deep Convolutional Networks Based on PCA

This chapter discusses the main motivations for using PCA-based deep convolutional networks—architectures that utilize several advantages of PCA and CNNs and add simplicities that make them interesting for use in visual object recognition and plant classification tasks. The architecture of the proposed network is explained in detail and required formulas are derived. This chapter also introduces the technique of spatial pyramid pooling and is finalized by an overview of the SVM classifier implemented in the final stage of our proposed architecture.

4.1 Background

Several groups within the computer vision and machine learning communities have proposed various modifications to the deep convolutional networks and have been able to empirically prove the superior performance of these networks [17]. However, ScatNets [18, 19] have been the first proposed systems among deep networks for which a mathematical proof has been presented. But ScatNet is a prefixed network; it does not have a learning capability as its convolutional filters are just wavelet operators. Although it has shown superior performance over CNNs in challenging texture recognition and handwritten digit tasks, its prefixed architecture has not generalized well to other object recognition problems suffering from large intra-class variations.

As the system proposed in [20], PCA network (PCANet) is a fusion of principal com-

ponent analysis and deep convolutional networks which applies PCA as an unsupervised learning method to the image patches to learn multi-stage filter banks. After learning the network weights, indexing and pooling are performed through simple binary hashing and normalized block histograms. Although PCANet uses the key concepts of DCNs, it seems to be a suboptimal solution which needs shorter learning time and fewer amounts of data for training.

An ancestor network closely related to PCANet was the two-stage oriented PCA (OPCA) first proposed for audio processing [70]. Although OPCA does not couple with hashing and local histogram extraction in the output layer, considering similar noise covariance, it obtains more robustness to noise and distortions compared to PCANet. However, PCANet has the same benefit of OPCA in providing more invariance for the intra-class variability.

4.2 Motivations

As already mentioned in Section 2.2, large intra-class variability is an inherent issue in plant identification problems; hence, ScatNet or similar prefixed deep networks cannot perform well in these tasks either, opening the way for exploring a deep learning approach for our problem of interest.

The literature has proven evidence that DCNs have had superior performance on large image databases such as ImageNet [71] and CIFAR [72]. However, they require a large number of layers to achieve good results that make their implementation computationally expensive. As a result, we believe an alternative must exist with a simpler baseline for learning network weights while having a reduced learning time that is able to achieve comparable results with those of the common CNNs.

PCA is an unsupervised learning method known for being advantageous in noise reduction, dimensionality reduction, and orientation normalization. Considering principal components (eigenvectors) as weights, we have been able to ignore the noise and find suboptimal filters with invariance properties. In addition, filters built using the principal components are essentially computed from data by minimizing reconstruction error and thus represent the learning capability while wavelet and Gabor filters have prefixed values. Therefore, it would be desirable to combine PCA features into a deep convolutional

network and to use all those benefits for plant identification or similar challenging tasks. These ideas allow us to modify PCANet and use it for our specific object recognition problem.

4.3 Contributions

We have improved the simple proposed baseline for convolutional networks using PCA filters and modified PCANet for our purpose of plant identification. The main improvements are achieved at the output layer where we apply max-pooling and spatial pyramid pooling. Moreover, we have extracted pyramid of features from unnormalized local histograms. These approaches provide pose invariance in addition to a reduction in overfitting. In addition, we utilize color information by learning filter banks from HSY color space which seems to be a simple color transform and more suitable for plant identification problem (see Appendix A).

4.4 Proposed Deep PCA Network

PCA network initializes by taking overlapping patches of all images and applying PCA to them to find filter banks by selecting principal components of the calculated eigenvectors. The obtained filters are then convolved with input images within the first layer of the convolutional network. Projections of patches on to the principal components form the responses of units in the first layer.

We can repeat this methodology for the obtained filtered images to compose the deep convolutional network architecture. That is to say, the next stage uses the same procedure to calculate and apply filters on the outputs of the first layer, forming a cascaded linear map. Next, the method uses binary quantization and hashing for multi-stage filtered image sets to concatenate them in the decimal form. Finally, local histograms are extracted as features from the blocks of the quantized images using the technique of spatial pyramid pooling.

The detailed explanation of this algorithm is as follows. The training data contains $i = 1, 2, \dots, N$ images I_i of size $m \times n$. In the first stage, patches of size $k_1 \times k_2$ pixels are extracted around each pixel in the image I_i . Afterwards, all such overlapping patches are

collected, vectorized, and mean subtracted to obtain X_i . Repeating this operation for all images, we obtain a patch collection X such that

$$X = [X_1, X_2, \dots, X_N] \in \mathbb{R}^{k_1 k_2 \times Nmn} \quad (4.1)$$

Next, in order to calculate the desired filter banks of orthonormal filters, V , PCA minimizes the reconstruction error to compute their L_1 principal components. The constrained optimization is formulated as

$$\min_{V \in \mathbb{R}^{k_1 k_2 \times L_1}} \|X - VV^T X\|_F^2 \quad \text{subject to} \quad V^T V = I_{L_1} \quad (4.2)$$

where $\|\cdot\|_F$ is the Frobenius norm, I_{L_1} is the identity matrix of size $L_1 \times L_1$ and the solution simply consists of finding L_1 principal eigenvectors of XX^T . Therefore, PCA filters for the first layer form weights $W_{l_1}^1$ for $l_1 = 1, 2, \dots, L_1$ by converting eigenvectors to matrices of size $k_1 \times k_2$. Hence, the l_1 -th filtered image is calculated by convolving the l_1 -th filter with the i -th patch-mean removed image, \bar{I}_i , as

$$I_i^{l_1} = \bar{I}_i * W_{l_1}^1 \quad (4.3)$$

We can repeat the same approach to learn L_2 PCA filters for the second layer to create double-filtered images. For this purpose, all the overlapping patches of each filtered image $I_i^{l_1}$ are collected, vectorized, and mean subtracted to obtain $Y_i^{l_1}$. Repeating this algorithm for all filtered images, we obtain,

$$Y = [Y_1^1, \dots, Y_N^1, Y_1^2, \dots, Y_N^2, \dots, Y_1^{L_1}, \dots, Y_N^{L_1}] \in \mathbb{R}^{k_1 k_2 \times L_1 Nmn} \quad (4.4)$$

Similarly, PCA filters for the second layer, $W_{l_2}^2$ for $l_2 = 1, 2, \dots, L_2$, are obtained by finding L_2 principal eigenvectors of YY^T and rearranging them as matrices of size $k_1 \times k_2$. Therefore, the double-filtered image maps, computed sequentially using the l_1 -th and l_2 -th filters, are obtained by convolving the l_2 -th filter with the i -th patch-mean removed filtered image, $\bar{I}_i^{l_1}$, as

$$O_i^{l_1, l_2} = \bar{I}_i^{l_1} * W_{l_2}^2 \quad (4.5)$$

As can be seen, in the output O for each image, we have $L_1 \times L_2$ double-filtered images with real values. To decrease the number of maps, we first binarize them using the Heaviside step function, $H(\cdot)$. Next, for each pixel, we map L_2 quantized binary bits to a decimal number as

$$T_i^{l_1} = \sum_{l_2=1}^{L_2} 2^{l_2-1} H(O_i^{l_1, l_2}) \quad (4.6)$$

This weighted sum acts similarly to a cell with prefixed weights in the descending order based on principality of the components. In fact, this conversion maps each L_2 binary bits acquired from corresponding pixels of the double-filtered binary images into a single graylevel image pixel in the range of $[0, 2^{L_2} - 1]$.

Later on, we apply max-pooling to discover the most responsive units from the obtained maps. This approach will provide a form of translation invariance and reduce the output map dimension to $\lfloor \frac{m}{s} \rfloor \times \lfloor \frac{n}{s} \rfloor$, where $s \geq 1$ is an integer that represents the subsampling step.

Finally, to build a more abstract representation of output maps, we use spatial pyramid pooling. We partition each of the subsampled decimal images, $S_i^{l_1}$, into p^2 blocks, where $p \geq 1$ is an integer indicating the pyramid level. Next, we compute block histograms (with 2^{L_2} bins) for all L_1 images as features of the i -th image, i.e.

$$f_i^p = \left[\text{hist}(B_i^{1,1}), \dots, \text{hist}(B_i^{1,2^p}), \dots, \text{hist}(B_i^{L_1,1}), \dots, \text{hist}(B_i^{L_1,2^p}) \right] \in \mathbb{R}^{1 \times 2^{L_2} 2^p L_1} \quad (4.7)$$

where $\text{hist}(B_i^{l_1, j})$ computes the histogram of the j -th block from the l_1 -th partitioned map of the i -th image. We can concatenate histograms from different levels of pyramids together to form the output feature vector for the i -th image. As previously discussed, using pyramids of local histograms provides invariance properties of large object poses and complex backgrounds. Figure 4.1 shows the block diagram of a two-stage PCA network.

We can extend the PCA filter learning to use it for color images in object databases. Weights are learned from each color channel in the same manner. Finally, we stack the weights together to obtain color filter banks and concatenate the extracted features from each color map to form a unit feature vector.

The patch-mean removal centers all the patches in the origin of the vector space.

This method is highly similar to the local contrast normalization in CNN [73]. We use binarization to obtain abstract representations of maps and utilize hashing to combine these quantized maps. Still, decimation is a simple way to unite output maps and one may find better weighing scales to combine maps. Moreover, the histogram offers some degrees of translation invariance to the extracted features; this is similar to the translation invariance obtained as a result of hand-crafted feature extraction methods such as SIFT [21], HOG [23], learned features such as the BoW model [74], and pooling processes in CNN [68, 69, 73, 75–77].

All the processes performed before the pooling layers of deep PCA network are completely linear. The nonlinearity process is only applied in the last layer and not between the stages. This is in contrast to the common approach of building deep neural networks such as the absolute rectification layer in CNN [73] and the modulus layer in Scat-Net [18, 19]. Our experiments show that using absolute rectification between all stages does not improve the final classification results. We believe the reason is that the use of quantization, max-pooling, and SPP in the output layer already introduces sufficient invariance and robustness in the final feature.

4.4.1 Spatial Pyramid Pooling

Spatial pyramid pooling (SPP) or spatial pyramid matching (SPM) is a feature pooling method that builds a more abstract representation of images. It preserves some of the spatial information by partitioning images into divisions from finer to coarser levels and aggregates local features in them [78, 79]. This higher-order representation is an extension of the Bag-of-Visual-Words (BoVW) model [80] which obtains a fixed-length output and can introduce invariance properties of objects poses including position, orientation, and scale.

SPP has a number of remarkable properties for deep convolutional networks. In contrast to the sliding window pooling which uses a single window size [81], SPP uses multi-level spatial bins and generates a fixed-length output regardless of the input size. Multi-level pooling has been shown to be robust to object deformations [79]. In addition, using a fixed number of bins instead of a fixed window size not only makes it possible to generate representations from arbitrarily sized images/windows for testing, but also allows us to feed images with varying sizes or scales during training. This flexibility, in turn,

increases the scale invariance and decreases overfitting. The aforementioned fixed-length vectors can then be used by the fully-connected neural network layers or classifiers such as SVM and Softmax.

4.4.2 Classification by Linear SVM

In the final stage, the multi-class linear support vector machine was chosen as the classifier to handle the complexity and accuracy issues due to the massive size of data obtained after feature pooling. For implementation, we use LIBLINEAR [82], an open source library for large-scale linear classification of binary and multi-class problems. It supports two popular supervised learning models of support vector machines (SVMs) and logistic regression (LR). Given a set of instance vectors with n -dimensional features $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, 2, \dots, m$, and their respective label vector $\mathbf{y} \in \mathbb{R}^m$ where $y_i \in \{-1, +1\}$, a linear classifier solves the following unconstrained optimization problem in its primal form to generate a weight vector \mathbf{w} for its model [83]

$$\min_{\mathbf{w}} f(\mathbf{w}) + C \sum_{i=1}^m \xi(\mathbf{w}, \mathbf{x}_i, y_i) \quad (4.8)$$

where $C > 0$ is the penalty parameter. The first term of the objective function, $f(\mathbf{w})$, is normally replaced by $\frac{1}{2} \mathbf{w}^T \mathbf{w}$, i.e. the L2-norm or regularization. In case a sparse solution for \mathbf{w} is required, the aforementioned term can be substituted with the L1-norm, indicated as $\|\mathbf{w}\|_1$. Also, the loss function term $\xi(\mathbf{w}, \mathbf{x}_i, y_i)$ can be defined as one of the following expressions

$$\xi(\mathbf{w}, \mathbf{x}_i, y_i) = \begin{cases} \max(0, (1 - y_i \mathbf{w}^T \mathbf{x}_i)), & \text{L1-loss SVC} \\ [\max(0, (1 - y_i \mathbf{w}^T \mathbf{x}_i))]^2, & \text{L2-loss SVC} \\ \max(0, (|y_i - \mathbf{w}^T \mathbf{x}_i| - \varepsilon)), & \text{L1-loss SVR} \\ [\max(0, (|y_i - \mathbf{w}^T \mathbf{x}_i| - \varepsilon))]^2, & \text{L2-loss SVR} \\ \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i)), & \text{Logistic Regression} \end{cases} \quad (4.9)$$

where $\varepsilon \geq 0$ specifies the loss sensitivity, and SVC and SVR indicate support vector classification and support vector regression options of SVM, respectively. Finally, the decision for each instance, d_i , is made based on the obtained weights using the sign

function

$$d_i = \text{sgn}(\mathbf{w}^T \mathbf{x}_i) \quad (4.10)$$

Therefore, the class of sample vector \mathbf{x}_i is predicted as positive if $d_i > 0$ and negative otherwise. Moreover, for multi-class problems, LIBLINEAR uses a one-vs-the-rest (OvR) or one-vs-all (OvA) strategy and a method by Crammer and Singer [84]. In other words, for a k -class classification purpose, the decision is made based on $\arg\max_j (\mathbf{w}_j^T \mathbf{x}_i)$, $j = 1, 2, \dots, k$. LIBLINEAR also allows the classifier to include a bias term. In this case, a constant is added as an extra variable to the sample vector \mathbf{x}_i and an extra column is concatenated to the output weights to represent the optimized bias vector.

As it can be seen, SVM is essentially a quadratic programming (QP) problem. This optimization problem can be solved either in the primal or in the dual forms. When using a nonlinear kernel to transform the original dataspace, the best option is to solve the dual problem in which the number of variables is equal to the number of training data instances. However, in the strictly linear case, we can solve the primal form—which is also a quadratic problem, but with an equal number of variables and data features as well as equal number of constraints and data samples. LIBLINEAR gives the option to solve either the primal or the dual forms with several variations such as L1/L2 regularization and L1/L2 loss without any kernel transform. Clearly, utilizing these options depends on the conditions of data at hand. For example, when the data is highly sparse or the number of features is much higher than the number of samples, it is more suitable to use the dual-based solvers with coordinate descent methods. In other cases, applying the primal-based solvers using Newton-type methods is preferable. As a result, LIBLINEAR is faster than LIBSVM [85] with a linear kernel.

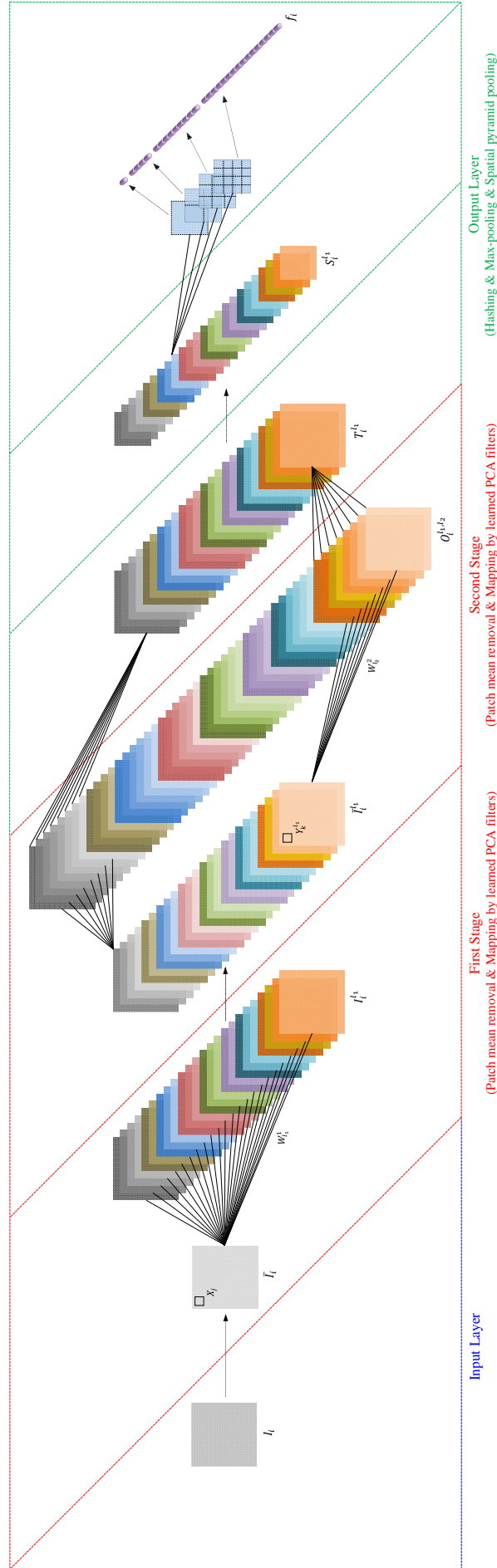


Figure 4.1: Block diagram of a two-stage PCA network

Chapter 5

Experiments and Results

In this chapter, we will evaluate performance of the proposed method and explain the utilized datasets and metrics. We will review several different results of our experiments to adjust optimal parameters of the proposed system. We will also see the effects of preprocessing on the performance. Next, we will show the results of evaluating robustness of our system against various variations on the test images. Finally, we will compare our system with those submitted to the LifeCLEF 2014 competition and discuss the time complexity issue.

5.1 Dataset Description

The core dataset we used to evaluate our proposed system is the one provided for plant identification task in LifeCLEF 2014 [13]. This dataset involves 500 species of trees, herbs, and ferns from photographs of their different organs mostly taken inside France by different users. The collected dataset contains 60,961 pictures in total, 47,815 images for training and 13,146 images for testing. Table 5.1 shows details of the provided datasets and their sample images.

Table 5.1: Details of plant identification datasets within the LifeCLEF 2014 [13]

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem
No. of Training Samples	1,987	6,356	13,164	3,753	7,754	11,335	3,466
No. of Test Samples	731	2,983	4,559	1,184	2,058	696	935
No. of Classes (Species)	356	490	483	374	470	212	388



Figure 5.1: Samples of LifeCLEF 2014 plant dataset [13]. Columns from left to right include Branch, Entire, Flower, Fruit, Leaf, LeafScan, and Stem, respectively.

The dataset contains meta-data for photos with the class label, species, genus, date, quality index (vote), location, photographer (user), etc. It is collected by different users in an observations-based manner to promote plant identification based on multi-image query. In other words, each photographer or author has used the same camera to take snapshots from different views of various organs of a plant species under similar lighting conditions on the same day. Finally, the test set is generated by randomly choosing a half of observations for each species. Users of selected photos for the test set are not necessarily the same as those for the training set—making the identification a real-world problem. Figure 5.1 shows samples of each category used for plant identification in LifeCLEF 2014.

5.2 Preprocessing

Before applying the proposed system, we modify original color images by transforming them from the RGB color space to HSY to acquire more intuitive and perceptual color in-

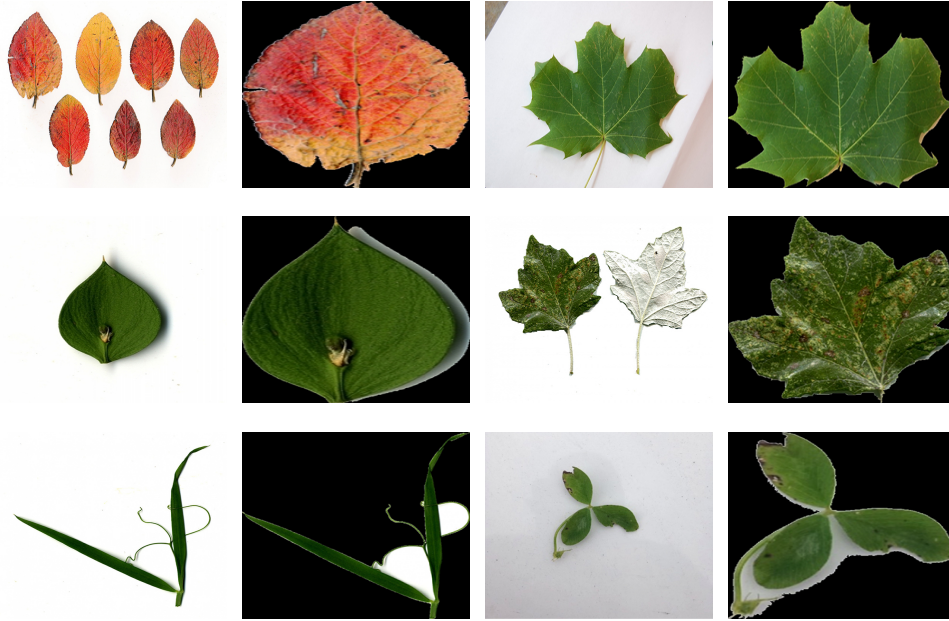


Figure 5.2: Samples of scanned leaves in the LifeCLEF 2014 plant dataset before and after preprocessing

formation. Moreover, for scanned leaves we apply a number of preprocessing techniques to improve the system performance. We utilize segmentation methods to discriminate at most one leaf (single connected component) from the background and next perform size and rotation normalization as in [9]. Generally, segmentation is achieved by applying a combination of morphological techniques followed by an adaptive thresholding for background removal. Next, we align major axis of leaves with the vertical axis and normalize all heights to 600 pixels to preserve the aspect ratio. After this size normalization, we use PCA and leaf petiole's location to perform orientation normalization. Figure 5.2 shows samples of scanned leaf images before and after preprocessing.

5.3 Evaluation Metrics

The first utilized metric for evaluating our results is the total classification accuracy. This metric calculates the accuracy (in percentage) as follows

$$Acc = \frac{N_C}{N_T} \times 100 \quad (5.1)$$

where N_T and N_C are the number of test samples and the number of correctly predicted class labels using the classifier, respectively. In addition, LifeCLEF itself employs a user-

based metric called the average inverse rank score [86] instead of the total classification accuracy. The average inverse rank score S is defined as

$$S = \frac{1}{U} \sum_{u=1}^U \frac{1}{P_u} \sum_{p=1}^{P_u} \frac{1}{N_{u,p}} \sum_{n=1}^{N_{u,p}} s_{u,p,n} \quad (5.2)$$

where U is the number of users who have taken the query pictures; P_u is the number of individual plants observed by the u -th user; $N_{u,p}$ is the number of pictures taken from the p -th plant observed by the u -th user; and $s_{u,p,n}$ is the inverse of the rank of the correct species for the given image, ranging from 0 to 1.

5.4 Experimental Methods

Before evaluating our system, we need to adjust available parameters of the proposed method. The main on-hand parameters of our deep PCA-based network are the number of learning stages, number of filters (nodes) in each stage, spatial pyramid levels, and the filtering patch size (receptive field size). Due to the combinatorial increase, each time we tune only one of the parameters until we find the optimal point.

After obtaining optimal points of all parameters, we evaluate our system performance by applying the system on the given dataset for plant identification. Finally, we conduct a number of experiments to measure the robustness of our system against pose and illumination variations.

5.4.1 Effects of Parameter Adjustment

To tune our system, we use the training set of preprocessed scanned leaf images. As Table 5.1 shows, the aforementioned dataset contains 11,335 images from isolated color leaves with 212 individual tree species. For parameter adjustment, we simply split the training dataset into two subsets: development set involving 9,532 images of all 212 unique classes, and validation set involving 1,803 images of 133 unique classes. We initialize our system by the following parameters and adjust them one by one until reaching optimal values. We normalize the size of input images to 64×32 pixels—quite small because of the computational complexity issue, and transform color images to the HSY color model. We choose 7×7 pixels for filtering patch size and use max-pooling with

subsampling step size of 2×2 pixels and a spatial pyramid in its fourth level.

5.4.1.1 Number of Learning Stages

We start with only one learning stage and increase the number of stages to see the effects of this variation on the performance. Table 5.2 shows the classification results for different number of learning stages. The number of filters in each stage is also indicated in the first row of the table.

Table 5.2: Classification results for different number of learning stages

No. of Stages / No. of Filters	1 / [8]	2 / [4,8]	3 / [4,4,8]	4 / [4,4,4,8]
Classification Accuracy	58.29%	67.61%	67.50%	67.39%
Inverse Rank Score	0.5237	0.6003	0.5916	0.5921

As the table shows, the best results are achieved by applying a two-stage convolutional PCA network.

5.4.1.2 Number of Filters

We use a two-stage network, initially fix the number of filters in the second stage at 8, and vary the number of filters in the first stage from 1 up to 48. Clearly, we have at most $7 \times 7 = 49$ filters in each stage and expect that the smallest principal components represent the noise information. Table 5.3 shows the classification results for various numbers of filters in the first stage.

Table 5.3: Classification results for different number of filters in the first stage

No. of Filters	[1,8]	[4,8]	[8,8]	[16,8]	[24,8]	[32,8]	[48,8]
Classification Accuracy	55.96%	67.61%	69.72%	71.99%	72.55%	72.49%	72.38%
Inverse Rank Score	0.4952	0.6003	0.6203	0.6369	0.6464	0.6416	0.6409

As can be seen, the best results are achieved by using 24 filters in the first stage of the two-stage network. We next fix the number of filters in the first stage at 24 and vary the number of filters in the second stage from 1 to 12. Table 5.4 shows the results for this experiment.

Table 5.4: Classification results for different number of filters in the second stage

No. of Filters	[24,1]	[24,4]	[24,8]	[24,10]	[24,12]
Classification Accuracy	54.52%	68.66%	72.55%	71.55%	69.27%
Inverse Rank Score	0.5145	0.6070	0.6464	0.6316	0.6093

As the results show, the best results are achieved by using 8 filters in the second stage of the two-stage network.

5.4.1.3 Filtering Patch Size

We now apply a two-stage network using 24 and 8 filters in the first and second stages, respectively. We perform an experiment by increasing the patch size from 5×5 up to 15×15 . Table 5.5 shows the results of this experiment.

Table 5.5: Classification results for different filtering patch sizes

Patch Size	[5,5]	[7,7]	[9,9]	[15,15]
Classification Accuracy	72.49%	72.55%	70.83%	68.50%
Inverse Rank Score	0.6408	0.6464	0.6252	0.5934

As we see, the best results are achieved by applying filters on the 7×7 patches of images.

5.4.1.4 Spatial Pyramid Levels

In this experiment, we use a two-stage PCA-based network with adjusted parameters obtained in the previous experiments. We initialize the experiment by taking the first level of spatial pyramid and pooling histogram from the total image block. Next, we increase the pyramid level up to the fifth level and pool histograms from 25 image blocks. Finally, we combine extracted features from different levels to construct the pyramid. Table 5.6 shows the results for this experiment.

As can be seen, the best results are achieved by pooling histograms from a spatial pyramid including the first four levels.

Table 5.6: Classification results for different levels of spatial pyramid

Pyramid Levels	[1]	[2]	[3]	[4]	[5]	[1,2,3,4]
Classification Accuracy	69.11%	71.99%	72.43%	72.55%	71.66%	72.66%
Inverse Rank Score	0.6031	0.6337	0.6363	0.6464	0.6267	0.6473

5.4.1.5 Image Size Normalization

Our last experiment tends to explore how varying the image size affects the system performance. Until now, we actually used a normalized image size to speed up the tuning procedure. In fact, this small image size conveys less information, hence calculated histograms would be sparse. This in turn will increase the speed of classification. Table 5.7 shows the classification results for various image sizes. In this experiment, we utilize only the fourth level of the spatial pyramid.

Table 5.7: Classification results for different image sizes

Image Size	[32,16]	[64,32]	[128,64]	Full Size
Classification Accuracy	65.50%	72.55%	77.20%	77.98%
Inverse Rank Score	0.5931	0.6464	0.6795	0.6832

As it can be seen, the best results are achieved by taking the original image size. However, there is a trade-off between the complexity and performance as we will discuss it later in this chapter.

5.4.2 Classification Results

After obtaining optimal values of the network parameters, we adjust our system to apply it to the different categories of LifeCLEF 2014 plant identification dataset. We utilize provided training and test sets of different categories for this experiment. Table 5.8 displays our obtained classification results for different categories of the aforementioned dataset.

As it can be seen, the scanned leaf, flower, and fruit photographs are relatively easier to classify compared to the stem, leaf, branch, and entire categories. All these categories except for the LeafScan show relatively fair results as their photographs include complex backgrounds with high amounts of intra-class variability in contrast to the isolated leaves.

Table 5.8: Classification results of the proposed method for different categories of plant identification task in LifeCLEF 2014

Category	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem
Classification Accuracy	16.42%	20.05%	26.76%	22.55%	21.33%	68.25%	24.92%
Inverse Rank Score	0.1561	0.1834	0.2677	0.2170	0.1928	0.6157	0.1679

Also, when we evaluate our system on the unprocessed LeafScan dataset, the classification accuracy decreases to 50.14% and its inverse rank score decreases to 0.4478. Hence, we can conclude that preprocessing is a major requirement prior to applying machine learning tasks.

Table 5.9 compares inverse rank scores of different systems submitted to the LifeCLEF 2014. As can be seen, our scheme scores the second place between all submissions. However, we remember that the winner of this competition has used several preprocessing techniques in addition to applying a CNN with 5 convolutional layers [49].

Table 5.9: Inverse rank scores of different systems submitted to the LifeCLEF 2014

	Branch	Entire	Flower	Fruit	Leaf	LeafScan	Stem
IBM Australia [49]	0.292	0.333	0.585	0.339	0.318	0.64	0.269
Proposed System	0.156	0.183	0.268	0.217	0.193	0.616	0.168
PlantNet [50]	0.112	0.167	0.366	0.197	0.165	0.541	0.152
Sabanci-Okan [87]	0.007	0.077	0.149	0.118	0.066	0.449	0.089
FINKI [88]	0.088	0.117	0.255	0.177	0.160	0.400	0.157
BME TMIT [51]	0.052	0.060	0.115	0.070	0.019	0.119	0.072
I3S [89]	0.041	0.023	0.040	0.040	0.035	0.089	0.086

5.4.3 Robustness of the Proposed System

To finalize our evaluations on the system performance, we conduct some experiments for robustness against pose and illumination variations. For this purpose, we use our learned system on the preprocessed scanned leaves as mentioned above, and apply it to the manipulated test images as in the following steps.

5.4.3.1 Scale Invariability

In this experiment, we scale the size of test images by different values ranging from $2/3$ to 2. Table 5.10 shows the classification results for this experiment.

Table 5.10: Classification results for different scales of test images

Test Image Scale	$2/3$	$4/5$	1	$4/3$	2
Classification Accuracy	50.86%	60.92%	68.25%	61.93%	40.37%
Inverse Rank Score	0.4819	0.5713	0.6157	0.5531	0.3495

5.4.3.2 Translation Invariability

In this experiment, we shift pixels of test images to either right or down by different values ranging from 0×0 to 16×16 pixels. Table 5.11 shows the classification results for this experiment.

Table 5.11: Classification results for different translation sizes of test images

Test Image Translation Size	[0,0]	[4,4]	[8,8]	[16,16]
Classification Accuracy	68.25%	64.22%	61.93%	56.61%
Inverse Rank Score	0.6157	0.5909	0.5763	0.5554

5.4.3.3 Rotation Invariability

In this experiment, we rotate test images by different values ranging from 0 to 180 degrees. Table 5.12 shows the classification results for this experiment.

Table 5.12: Classification results for different rotation angles of test images

Test Image Rotation Angle	0°	10°	30°	90°	180°
Classification Accuracy	68.25%	38.65%	20.98%	23.56%	28.16%
Inverse Rank Score	0.6157	0.4057	0.2105	0.2514	0.2710

Considering these results, one may conclude that there is no good orientation invariance in the system. However, we note that the LeafScan dataset almost includes upright

objects (leaves) and the weights are learned only based on this orientation. Besides, we only apply PCA on the local patches of image and not the whole block. Therefore, we may improve the system robustness against rotation by augmenting training data using rotated images to reduce overfitting.

5.4.3.4 Illumination Invariability

In this experiment, we darken or brighten the test images by adding different values ranging from -50 to +50 to all pixel values of color channels. We next quantize the obtained values in the original dynamic range. Table 5.13 shows the classification results for this experiment.

Table 5.13: Classification results for different intensities of test images

Test Image Added Pixel Value	-50	-25	0	+25	+50
Classification Accuracy	45.69%	59.05%	68.25%	66.81%	61.21%
Inverse Rank Score	0.4734	0.5591	0.6157	0.5876	0.5468

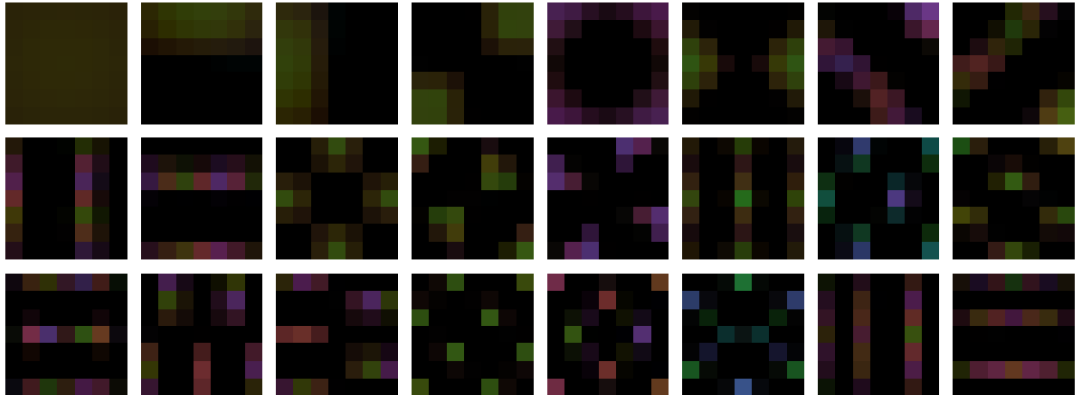
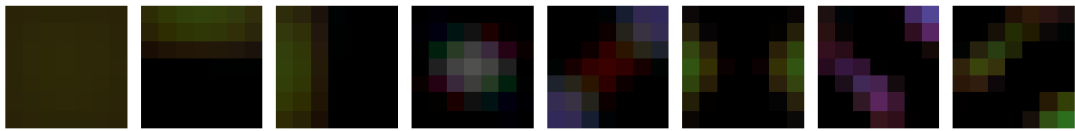
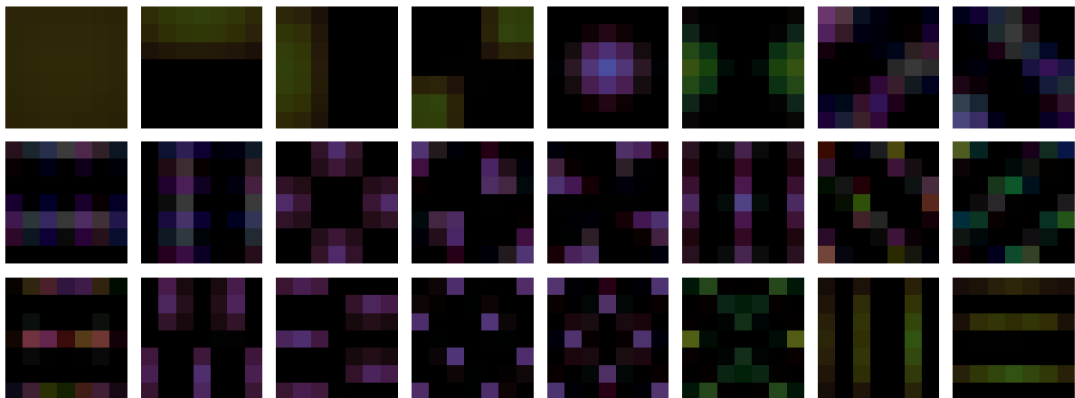
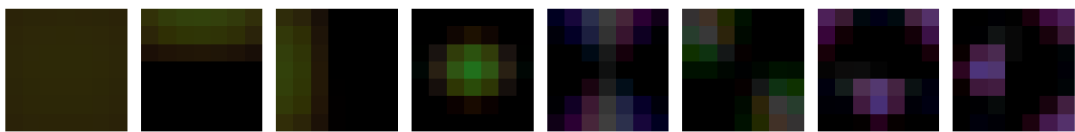
5.5 Time Complexity

We finally measured the complexity of our system in terms of the running time for feature extraction and classifier training. On average over all categories, the proposed deep PCA network took 1.63 seconds/image and 6.54 seconds/image for feature extraction and training, respectively. All codes were implemented in MATLAB (run in a 80 GB RAM and 2.50 GHz CPU with two processors).

5.6 Learned Filter Banks

As the final remark, we will present all weights learned by our proposed method during the process that led to obtaining results of Section 5.4.2. Figures 5.3 to 5.18 show learned weights using principal component analysis for different categories.

As we note, each learned filter or node in the network is composed of a 7×7 HSY color patch. In practice, each filter patch in the first stage is convolved with the image.

Figure 5.3: Learned weights from the Branch category in the 1st stage of PCA networkFigure 5.4: Learned weights from the Branch category in the 2nd stage of PCA networkFigure 5.5: Learned weights from the Entire category in the 1st stage of PCA networkFigure 5.6: Learned weights from the Entire category in the 2nd stage of PCA network

The obtained maps are then convolved with the filters in the second stage and their combination determines the output activations. In general, we can say that those image patches that are similar to the first stage filter banks are more likely to be activated. Subsequently, those patches of the obtained maps or filtered images that are close to the second stage filter banks in pattern and color (brightness, hue, and saturation) will actually become activated.

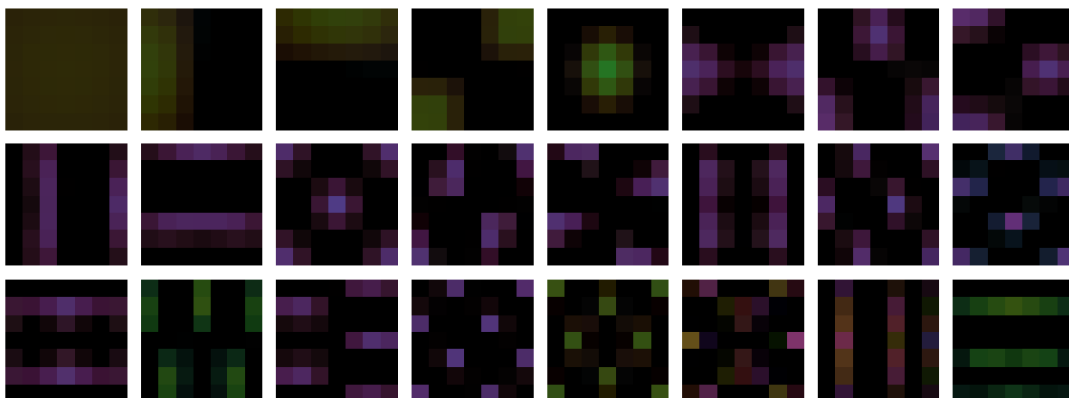


Figure 5.7: Learned weights from the Flower category in the 1st stage of PCA network

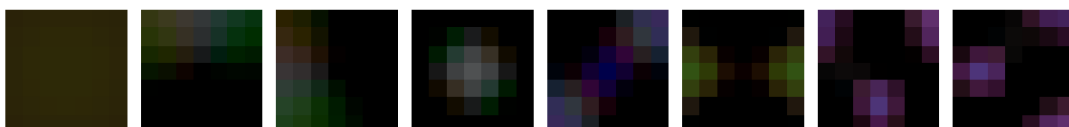


Figure 5.8: Learned weights from the Flower category in the 2nd stage of PCA network

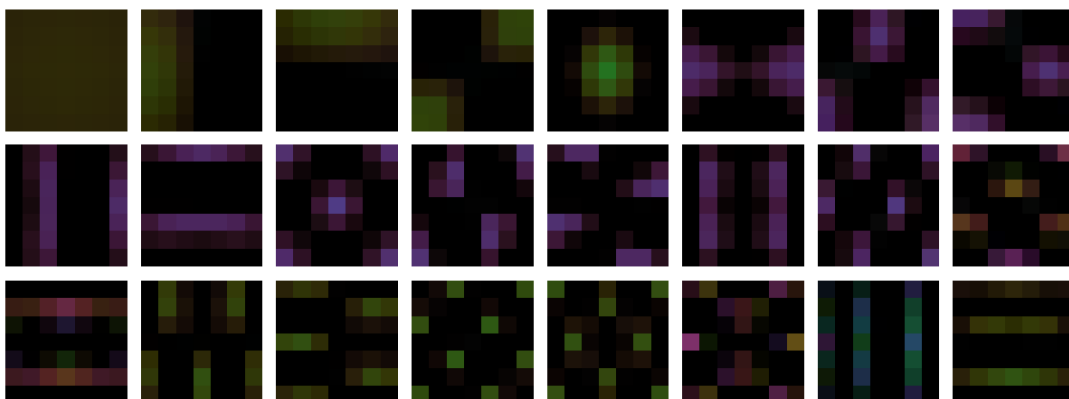


Figure 5.9: Learned weights from the Fruit category in the 1st stage of PCA network

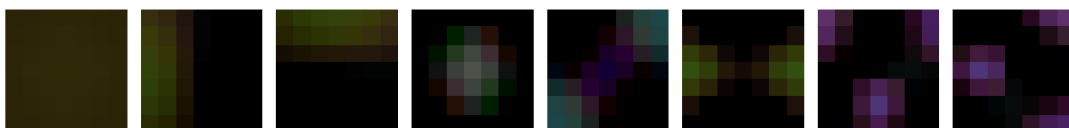


Figure 5.10: Learned weights from the Fruit category in the 2nd stage of PCA network

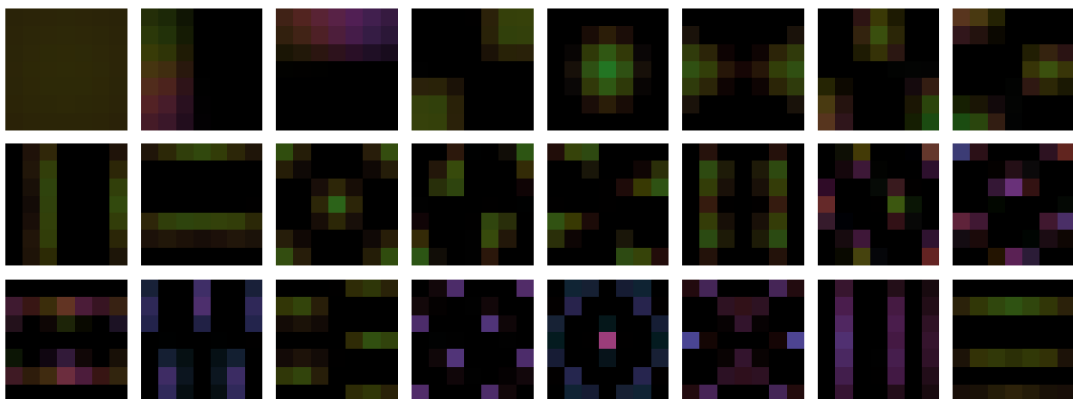


Figure 5.11: Learned weights from the Leaf category in the 1st stage of PCA network

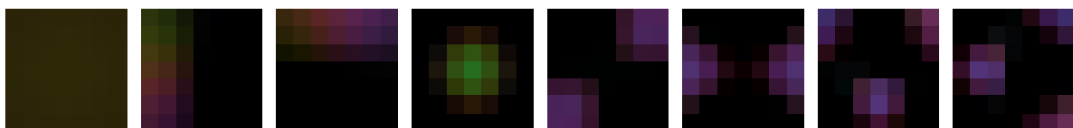


Figure 5.12: Learned weights from the Leaf category in the 2nd stage of PCA network

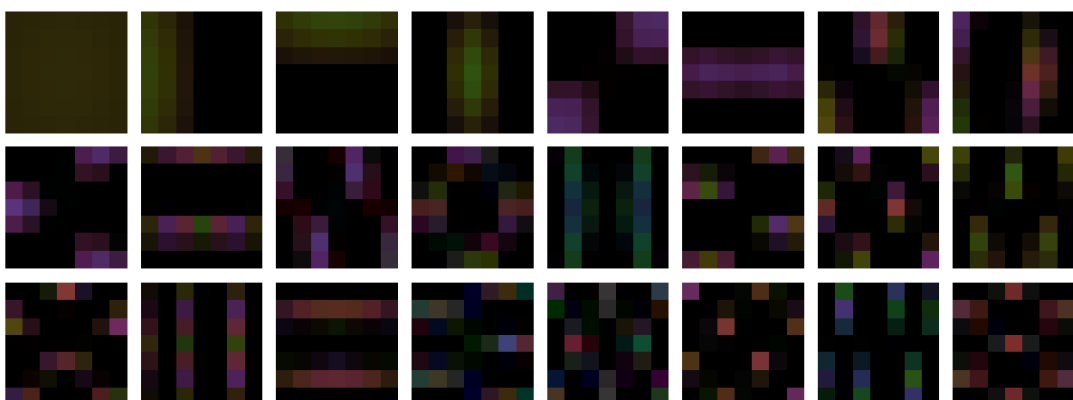


Figure 5.13: Learned weights from the LeafScan category in the 1st stage of PCA network

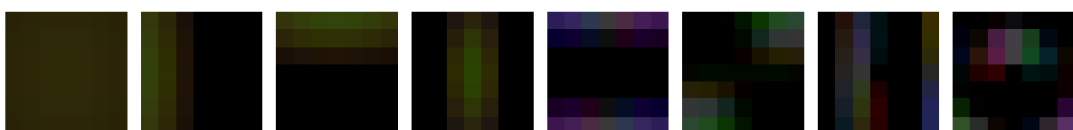


Figure 5.14: Learned weights from the LeafScan category in the 2nd stage of PCA network

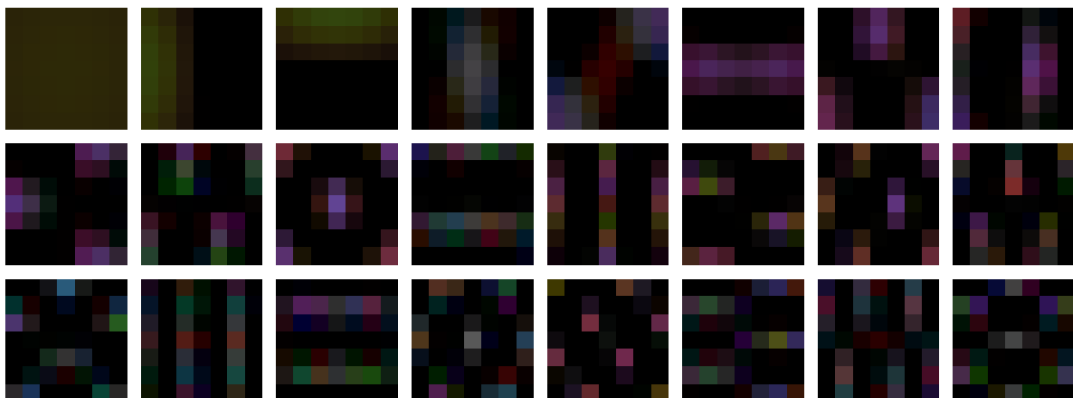


Figure 5.15: Learned weights from the preprocessed LeafScan category in the 1st stage of PCA network

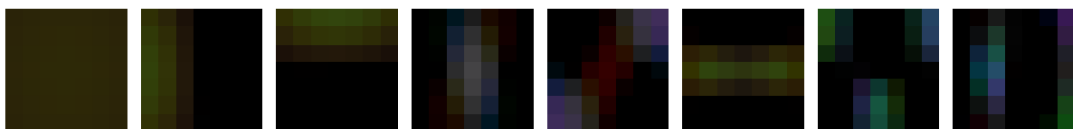


Figure 5.16: Learned weights from the preprocessed LeafScan category in the 2nd stage of PCA network

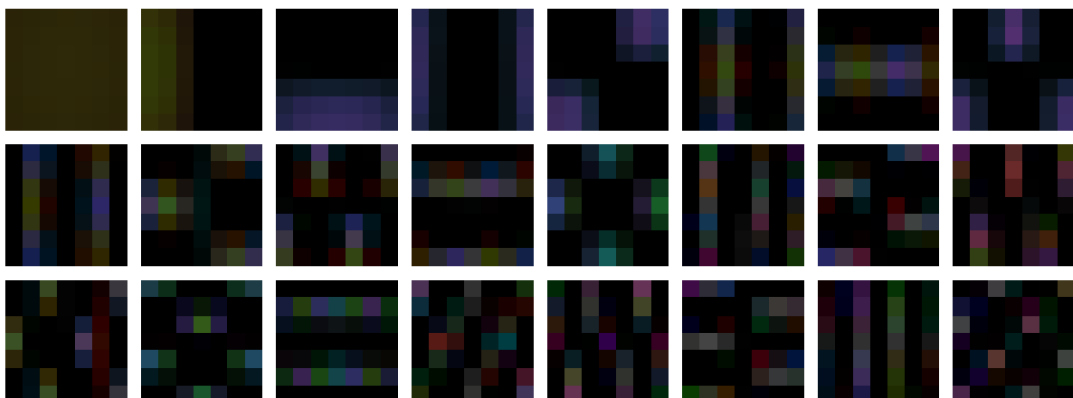


Figure 5.17: Learned weights from the Stem category in the 1st stage of PCA network

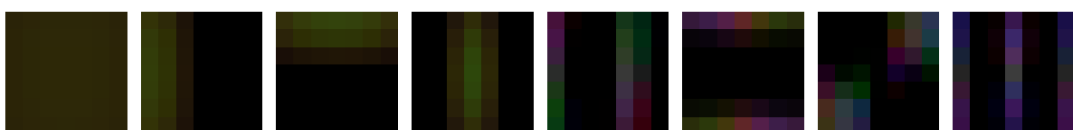


Figure 5.18: Learned weights from the Stem category in the 2nd stage of PCA network

Chapter 6

Summary and Conclusion

In this work, we proposed a deep convolutional architecture based on PCA network in order to identify plant species. We used the LifeCLEF 2014 dataset of the plant identification competition and evaluated our proposed approach by comparing the system performance for different plant categories with the best submitted systems in the same competition. This comparison indicated that our proposed system would have achieved the second place in almost all plant categories. Our best achieved performance was from the scanned leaves category with a classification accuracy of 68.25% and an inverse rank score of 0.6157.

The main strength of our work relies on the simplicity of the proposed network. In comparison with the winner of the competition which had applied a five-layer convolutional network along with a number of preprocessing techniques, we only used a two-stage convolutional network in an unsupervised learning way. Also, we only applied preprocessing for scanned leaf images. However, the system performance decreased when we removed the preprocessing step, resulting in a classification accuracy of 50.14% and an inverse rank score of 0.4478 for original scanned leaves.

Moreover, our results showed that the proposed system has the strength of being robust against small changes in translation, scaling, and illumination due to utilizing pooling schemes. However, the system is weak against rotation variations. One possible explanation for this weakness is that the almost upright orientations of leaves in the LeafScan dataset could be causing overfitting during the learning process. Since we apply PCA to the patches of such images, the system is able to correctly classify almost upright oriented leaves but it is less likely to classify leaves with rotated midribs. We may improve

the system's robustness against rotation by data augmentation using rotated leaves during training.

To sum up, this system is a simple baseline for deep learning that uses principal component analysis. Our system is highly robust and reliable for producing acceptable results in a time limited paradigm for object recognition tasks such as plant identification. Possible improvements could include combining our system with novel CNN methods or applying Gabor filters in a way similar to PCA to extract more low-level features from the data. Another improvement could be obtained by performing smart preprocessing techniques for removing the background and ineffectual plant elements.

Appendix A

HSY Color Space

The HSY color model [90] is a hue–saturation–brightness color system inspired by its family color models such as HSV (hue–saturation–value), HSI (hue–saturation–intensity), and HSL (hue–saturation–luminance). However, unlike the HSV and HSL models, HSY uses RGB components to compute a true luminance which considers human perception of colors in inferring different levels of brightness. In contrast to the RGB model, HSY encodes perceptually similar colors such as red and pink close to each other as they similar hue values. This makes HSY color model more suitable for color segmentation. Moreover, although there are many other color representations such as CIELAB, HSY gives an easy definition of saturation–independent from the brightness–which speeds up computations.

In the RGB space, colors are represented as vectors such as $[R, G, B]$ where the values of R , G , and B are within $[0, 1]$. But the transformation of the RGB color space to the HSY space is really converting from the rectangular coordinate system to the cylindrical one. Basically, a new axis is placed between the two points $[0, 0, 0]$ and $[1, 1, 1]$ and the color values are written in terms of cylindrical coordinates from this axis. Since all the gray or achromatic points with the property that $R = G = B$ belong to this axis, it is known as the achromatic axis. On this cylindrical-coordinate axis, brightness, hue, and saturation correspond to the color, angular coordinate, and distance from the achromatic axis, respectively.

Considering these explanations, the luminance and saturation coordinates are calcu-

lated as follows

$$Y = 0.299R + 0.587G + 0.114B \quad (\text{A.1})$$

$$S = \max(R, G, B) - \min(R, G, B) \quad (\text{A.2})$$

where Y and $S \in [0, 1]$. Assuming that C_1 and C_2 are components of the chroma, C , the value of hue is found using the following formulas

$$C_1 = R - \frac{1}{2}(G + B) \quad (\text{A.3})$$

$$C_2 = -\frac{\sqrt{3}}{2}(G - B) \quad (\text{A.4})$$

$$C = \sqrt{C_1^2 + C_2^2} \quad (\text{A.5})$$

and hence

$$H = \begin{cases} 360^\circ - \cos^{-1}\left(\frac{C_1}{C}\right), & \text{if } B > G \\ \cos^{-1}\left(\frac{C_1}{C}\right), & \text{otherwise} \end{cases} \quad (\text{A.6})$$

where $C \in [0, 1]$ and $H \in [0^\circ, 360^\circ]$. Having the values of HSY coordinates, one can easily reverse these transforms to obtain the RGB values.

Appendix B

Acronyms

ANN Artificial Neural Network

BoVW Bag of Visual Words

BoW Bag of Words

CBIR Content-Based Image Retrieval

CID Charge Injection Device

CLEF Conference and Labs of the Evaluation Forum

CNN Convolutional Neural Network

CNS Central Nervous System

ConvNet Convolutional Neural Network

DBN Deep Belief Network

DCN Deep Convolutional Network

DNN Deep Neural Network

EFA Elliptic Fourier Analysis

EFD Elliptic Fourier Descriptor

EOH Edge Orientation Histogram

GMM Gaussian Mixture Model

HOG Histograms of Oriented Gradients

ICA Independent Component Analysis

LBP Local Binary Pattern

LDA Linear Discriminant Analysis

LR Logistic Regression

MBR Minimum Bounding Rectangle

MF Morphological Feature

MMC Moving Median Center

MRF Markov Random Field

NN Nearest Neighbors

OvA One vs. All

OvR One vs. the Rest

PCA Principal Component Analysis

PCANet PCA Network

PDA Personal Digital Assistant

QP Quadratic Programming

RBF Radial Basis Function

ReLU Rectified Linear Unit

ROI Region of Interest

ScatNet Scattering Network

SGD Stochastic Gradient Descent

SIFT Scale-Invariant Feature Transform

SPM Spatial Pyramid Matching

SPP Spatial Pyramid Pooling

SURF Speeded-Up Robust Features

SVC Support Vector Classification

SVM Support Vector Machine

SVR Support Vector Regression

Bibliography

- [1] N. K. Logothetis and D. L. Sheinberg, “Visual object recognition,” *Annual review of neuroscience*, vol. 19, no. 1, pp. 577–621, 1996.
- [2] D. G. Lowe, “Three-dimensional object recognition from single two-dimensional images,” *Artificial Intelligence*, pp. 355–395, 1987.
- [3] M. Riesenhuber and T. Poggio, “Models of object recognition,” *Nature neuroscience*, vol. 3, pp. 1199–1204, 2000.
- [4] T. Gevers and A. W. M. Smeulders, “Color based object recognition,” in *International Conference on Image Analysis and Processing*, 1997, pp. 319–326.
- [5] S. K. Nayar and R. M. Bolle, “Reflectance based object recognition,” *International Journal of Computer Vision*, pp. 219–240, 1996.
- [6] A. R. Pope, “Model-based object recognition a survey of recent research,” *Technical Report*, 1994.
- [7] C.-L. Lee and S.-Y. Chen, “Classification of leaf images,” *International Journal of Imaging Systems and Technology*, pp. 15–23, 2006.
- [8] C. Tirkaz, D. Bruckner, G. Yin, and J. Haase, “Activity recognition using a hierarchical model,” in *38th Annual Conference on IEEE Industrial Electronics Society*, 2012, pp. 2814–2820.
- [9] B. Yanikoglu, E. Aptoula, and C. Tirkaz, “Automatic plant identification from photographs,” *Machine Vision and Applications*, vol. 25, no. 6, pp. 1369–1383, 2014.
- [10] H. Goëau, P. Bonnet, A. Joly, N. Boujemaa, D. Barthelemy, J.-F. Molino, P. Birnbaum, E. Mouysset, and M. Picard, “The CLEF 2011 plant images classification task,” in *CLEF (Notebook Papers/Labs/Workshop)*, Amsterdam, 2011.

- [11] H. Goëau, P. Bonnet, A. Joly, I. Yahiaoui, D. Barthelemy, N. Boujemaa, and J.-F. Molino, “The ImageCLEF 2012 plant identification task,” in *CLEF (Online Working Notes/Labs/Workshop)*, Rome, 2012.
- [12] H. Goëau, P. Bonnet, A. Joly, V. Bakic, D. Barthelemy, N. Boujemaa, and J.-F. Molino, “The ImageCLEF 2013 plant identification task,” in *CLEF (Working Notes)*, Valencia, 2013.
- [13] H. Goëau, A. Joly, P. Bonnet, S. Selmi, J.-F. Molino, D. Barthelemy, and N. Boujemaa, “LifeCLEF plant identification task 2014,” in *CLEF (Working Notes)*, Sheffield, 2014, pp. 598–615.
- [14] A. Joly, H. Goëau, C. Spampinato, P. Bonnet, W.-P. Vellinga, R. Planqué, A. Rauber, S. Palazzo, B. Fisher, and H. Müller, “LifeCLEF 2015: multimedia life species identification challenges,” in *CLEF 2015 Proceedings*, ser. Springer LNCS, 2015.
- [15] T. Kato, “Database architecture for content-based image retrieval,” in *SPIE/IS&T Symposium on Electronic Imaging: Science and Technology*, 1992, pp. 112–123.
- [16] B. Yanikoglu, E. Aptoula, and C. Tirkaz, “Sabanci-Okan system at ImageClef 2012: Combining features and classifiers for plant identification,” in *CLEF (Online Working Notes/Labs/Workshop)*, 2012.
- [17] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, pp. 436–444, 2015.
- [18] L. Sifre and S. Mallat, “Rotation, scaling and deformation invariant scattering for texture discrimination,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1233–1240.
- [19] J. Bruna and S. Mallat, “Invariant scattering convolution networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1872–1886, 2013.
- [20] T.-H. Chan, K. Jia, S. Gao, J. Lu, Z. Zeng, and Y. Ma, “PCANet: A simple deep learning baseline for image classification?” *Computing Research Repository (CoRR)*, 2014, arXiv:1404.3606v2.
- [21] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, pp. 91–110, 2004.

- [22] K. Grauman and B. Leibe, “Visual object recognition,” in *Visual Object Recognition*, 2011.
- [23] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.
- [24] D. Guyer, G. Miles, M. Schreiber, O. Mitchell, and V. Vanderbilt, “Machine vision and image processing for plant identification,” *Transactions of the ASAE*, vol. 29, no. 6, pp. 1500–1507, 1986.
- [25] J. S. Cope, D. P. A. Corney, J. Y. Clark, P. Remagnino, and P. Wilkin, “Plant species identification using digital morphometrics: A review,” 2012, pp. 7562–7573.
- [26] C. Tirkaz, J. Eisenstein, T. M. Sezgin, and B. Yanikoglu, “Identifying visual attributes for object recognition from text and taxonomy,” *Computer Vision and Image Understanding*, pp. 12–23, 2015.
- [27] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.
- [28] T. F. Stuessy, *Principles and practice of plant taxonomy*. Cambridge University Press, 2006.
- [29] J.-X. Du, X. Wang, and G.-J. Zhang, “Leaf shape based plant species recognition,” *Applied Mathematics and Computation*, pp. 883–893, 2007.
- [30] C.-C. Chen, “Improved moment invariants for shape discrimination,” *Pattern Recognition*, pp. 683–686, 1993.
- [31] F.-Y. Lin, C.-H. Zheng, X.-F. Wang, and Q.-K. Man, “Multiple classification of plant leaves based on gabor transform and lbp operator,” in *Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques*. Springer, 2008, pp. 432–439.
- [32] L. Ye and E. J. Keogh, “Time series shapelets: a new primitive for data mining,” in *KDD*, 2009, pp. 947–956.

- [33] F. Mokhtarian and S. Abbasi, "Matching shapes with self-intersections: Application to leaf classification," *IEEE Transactions on Image Processing*, vol. 13, no. 5, pp. 653–661, 2004.
- [34] R. J. White, H. C. Prentice, and T. Verwijst, "Automated image acquisition and morphometric description," *Canadian Journal of Botany*, vol. 66, no. 3, pp. 450–459, 1988.
- [35] J.-X. Du, D. Huang, X. Wang, and X. Gu, "Shape recognition based on radial basis probabilistic neural network and application to plant species identification," in *International Society of Nutrigenetics/Nutrigenomics (ISNN)*, 2005, pp. 281–285.
- [36] J. C. Neto, G. E. Meyer, D. D. Jones, and A. K. Samal, "Plant species identification using elliptic fourier leaf shape analysis," *Computers and electronics in agriculture*, vol. 50, no. 2, pp. 121–134, 2006.
- [37] B. Yanikoglu, E. Aptoula, and S. T. Yildiran, "Sabanci-Okan system at ImageClef 2013 plant identification competition," in *CLEF (Working Notes)*, 2013.
- [38] O. M. Bruno, R. de Oliveira Plotze, M. Falvo, and M. de Castro, "Fractal dimension applied to plant identification," *Information Sciences*, pp. 2722–2733, 2008.
- [39] R. de Oliveira Plotze, M. Falvo, J. G. Pádua, L. C. Bernacci, M. L. C. Vieira, G. C. X. Oliveira, and O. M. Bruno, "Leaf shape analysis using the multiscale minkowski fractal dimension, a new morphometric method: a study with *Passiflora* (Passifloraceae)," *Canadian Journal of Botany*, vol. 83, no. 3, pp. 287–301, 2005.
- [40] J. Clarke, S. Barman, P. Remagnino, K. Bailey, D. Kirkup, S. Mayo, and P. Wilkin, "Venation pattern analysis of leaf images," in *International Symposium on Visual Computing*, 2006, pp. 427–436.
- [41] Y. Li, Z. Chi, and D. D. Feng, "Leaf vein extraction using independent component analysis," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2006, pp. 3890–3894.
- [42] J. S. Cope, P. Remagnino, S. Barman, and P. Wilkin, "The extraction of venation from leaf images by evolved vein classifiers and ant colony algorithms," in *Advanced Concepts for Intelligent Vision Systems*, 2010, pp. 135–144.

- [43] M.-E. Nilsback and A. Zisserman, “Delving into the whorl of flower segmentation,” in *BMVC*, 2007, pp. 1–10.
- [44] A.-X. Hong, G. Chen, J. li Li, Z. ru Chi, and D. Zhang, “A flower image retrieval method based on roi feature,” *Journal of Zhejiang University Science*, vol. 5, no. 7, pp. 764–772, 2004.
- [45] T. Beghin, J. S. Cope, P. Remagnino, and S. Barman, “Shape and texture based plant leaf classification,” in *Advanced Concepts for Intelligent Vision Systems*, 2010, pp. 345–353.
- [46] A. N. Hussein, S. Mashohor, and M. I. Saripan, “A texture-based approach for content based image retrieval system for plant leaves images,” in *IEEE 7th International Colloquium on Signal Processing and its Applications (CSPA)*, 2011, pp. 11–14.
- [47] V. Bakic, I. Yahiaoui, S. Mouine, S. L. Ouertani, W. Ouertani, A. Verroust-Blondet, H. Goëau, and A. Joly, “Inria IMEDIA2’s participation at ImageCLEF 2012 plant identification task,” in *CLEF (Online Working Notes/Labs/Workshop)*, 2012.
- [48] S. Paris, X. Halkias, and H. Glotin, “Participation of LSIS/DYNI to ImageCLEF 2012 plant images classification task,” in *CLEF (Online Working Notes/Labs/Workshop)*, 2012.
- [49] Q. Chen, M. Abedini, R. Garnavi, and X. Liang, “IBM research Australia at LifeCLEF 2014: Plant identification task,” in *CLEF (Working Notes)*, 2014, pp. 693–704.
- [50] H. Goëau, A. Joly, I. Yahiaoui, V. Bakić, A. Verroust-Blondet, P. Bonnet, D. Barthelemy, N. Boujemaa, and J.-F. Molino, “Pl@ntNet’s participation at LifeCLEF 2014 plant identification task,” in *CLEF (Working Notes)*, 2014, pp. 724–737.
- [51] G. Szücs, D. Papp, and D. Lovas, “Viewpoints combined classification method in image-based plant identification task,” in *CLEF (Working Notes)*, 2014, pp. 763–770.
- [52] W. S. McCulloch and W. Pitts, “Neurocomputing: Foundations of research.” MIT Press, 1988, ch. A Logical Calculus of the Ideas Immanent in Nervous Activity, pp. 15–27.

- [53] D. O. Hebb, *The Organization of Behavior*. John Wiley, New York, 1949.
- [54] B. G. Farley and W. A. Clark, "Simulation of self-organizing systems by digital computer," *Transactions of the IRE Professional Group on Information Theory (TIT)*, pp. 76–84, 1954.
- [55] N. Rochester, J. H. Holland, L. H. Haibt, and W. L. Duda, "Tests on a cell assembly theory of the action of the brain, using a large digital computer," *Transactions of the IRE Professional Group on Information Theory (TIT)*, pp. 80–93, 1956.
- [56] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, p. 386, 1958.
- [57] M. Minsky and S. Papert, *Perceptrons - an introduction to computational geometry*. MIT Press, 1987.
- [58] A. H. Klopff, "Brain function and adaptive systems - a heterostatic theory," in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, 1974.
- [59] P. J. Werbos, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," Ph.D. dissertation, Harvard University, 1974.
- [60] B. Blaus. (2013) Anatomy of a multipolar neuron. [Online]. Available: https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png
- [61] S. C. Turaga, J. F. Murray, V. Jain, F. Roth, M. Helmstaedter, K. L. Briggman, W. Denk, and H. S. Seung, "Convolutional networks can learn to generate affinity graphs for image segmentation," *Neural Computation*, pp. 511–538, 2010.
- [62] F. Ning, D. Delhomme, Y. LeCun, F. Piano, L. Bottou, and P. E. Barbano, "Toward automatic phenotyping of developing embryos from videos," *IEEE Transactions on Image Processing*, pp. 1360–1371, 2005.
- [63] D. C. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, pp. 333–338, 2012.
- [64] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, pp. 1929–1958, 2014.

- [65] D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat's visual cortex," *The Journal of physiology*, vol. 160, no. 1, p. 106, 1962.
- [66] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, pp. 455–469, 1982.
- [67] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," *Computing Research Repository (CoRR)*, 2013, arXiv:1301.3557.
- [68] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [69] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Neural Information Processing Systems (NIPS)*, 2012, pp. 1106–1114.
- [70] C. J. C. Burges, J. C. Platt, and S. Jana, "Distortion discriminant analysis for audio fingerprinting," *IEEE Transactions on Speech and Audio Processing*, pp. 165–174, 2003.
- [71] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, pp. 1–42, 2015.
- [72] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Master's thesis, Department of Computer Science, University of Toronto, 2009.
- [73] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *IEEE International Conference on Computer Vision (ICCV)*, 2009, pp. 2146–2153.

- [74] F.-F. Li and P. Perona, “A bayesian hierarchical model for learning natural scene categories,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 524–531.
- [75] K. Kavukcuoglu, P. Sermanet, Y.-L. Boureau, K. Gregor, M. Mathieu, and Y. LeCun, “Learning convolutional feature hierarchies for visual recognition,” in *Neural Information Processing Systems (NIPS)*, 2010, pp. 1090–1098.
- [76] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, “Max-out networks,” in *International Conference on Machine Learning (ICML)*, 2013, pp. 1319–1327.
- [77] K. He, X. Zhang, S. Ren, and J. Sun, “Spatial pyramid pooling in deep convolutional networks for visual recognition,” in *European Conference on Computer Vision*, 2014, pp. 346–361.
- [78] K. Grauman and T. Darrell, “The pyramid match kernel: Discriminative classification with sets of image features,” in *IEEE International Conference on Computer Vision (ICCV)*, 2005, pp. 1458–1465.
- [79] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006, pp. 2169–2178.
- [80] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos,” in *IEEE International Conference on Computer Vision (ICCV)*, 2003, pp. 1470–1477.
- [81] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural Computation*, pp. 541–551, 1989.
- [82] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

-
- [83] C.-J. Lin, R. C. Weng, and S. S. Keerthi, “Trust region newton methods for large-scale logistic regression,” in *International Conference on Machine Learning (ICML)*, 2007, pp. 561–568.
- [84] K. Crammer and Y. Singer, “On the learnability and design of output codes for multiclass problems,” *Machine Learning*, pp. 201–233, 2002.
- [85] C.-C. Chang and C.-J. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, p. 27, 2011.
- [86] H. Müller, P. Clough, T. Deselarsers, and B. Caputo, *ImageCLEF: Experimental evaluation in visual information retrieval*, ser. The Information Retrieval Series. Springer, 2010, vol. 32.
- [87] B. Yanikoglu, S. T. Yildiran, C. Tirkaz, and E. Aptoula, “Sabanci-Okan system at LifeCLEF 2014 plant identification competition,” in *CLEF (Working Notes)*, 2014, pp. 771–777.
- [88] I. Dimitrovski, G. Madjarov, P. Lameski, and D. Kocev, “Maestra at LifeCLEF 2014 plant task: Plant identification using visual data,” in *CLEF (Working Notes)*, 2014, pp. 705–714.
- [89] M. Issolah, D. Lingrand, and F. Precioso, “Plant species recognition using Bag-Of-Word with SVM classifier in the context of the LifeCLEF challenge,” in *CLEF (Working Notes)*, 2014, pp. 738–746.
- [90] A. Hanbury, “A 3D-polar coordinate colour representation well adapted to image analysis,” in *Proceedings of the 13th Scandinavian conference on Image analysis*, 2003, pp. 804–811.